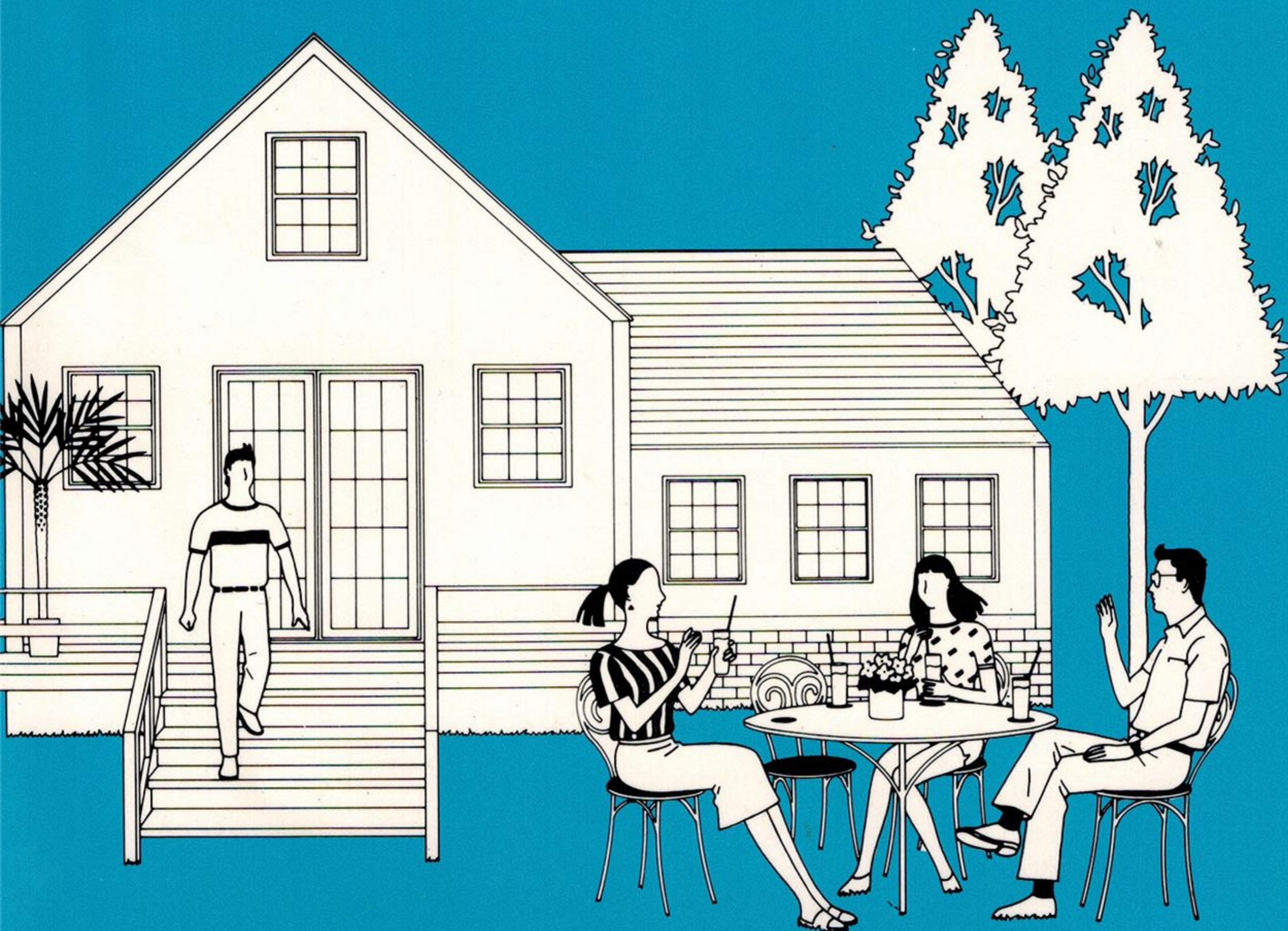
NECパーソナルコンピュータ
PC-9800シリーズ

PC-9801UV21

BASICプログラミング入門

PC-9801UV21-BM

ご注意

(1) 本書の内容の一部または全部を無断転載することは禁止されています.

(2) 本書の内容に関しては将来予告なしに変更することがあります.

(3) 本書は内容について万全を期して作成いたしましたが,万一御不審な点や誤り,記載もれなどお気付きのことがありましたら御連絡下さい.

(4) 運用した結果の影響について(3)項にかかわらず責任を負いかねますので御了承下さい.

BASICプログラミング入門

はじめに

8ビットマイクロプロセッサで幕を開けたパソコンも今や16ビットパソコンの実用段階を迎えております.

PC—9800シリーズは，本格的な16ビットパソコンとして，数々の特長ある強力な機能を提供します．例えば更に強化されたN_{88}—BASIC(86)，そして日本語処理や図形処理，また，ビジネスユースのための大容量ファイルの提供などがあげられます.

一方，従来のPCシリーズの大きな特長の一つである取り扱いの容易さは，そのまま受けつがれておりますので，新しい機能が提供されているからといって何もしり込みをすることはありません.

それよりも新たな挑戦意欲がわいてくるのではありませんか.

さて，本書は，BASIC言語を使ってPC—9800シリーズを使いこなしていくための，“プログラミングの手ほどき”として編集され，本書を読む上での特別な知識がなくても勉強できるように配慮しております.

ただ「プログラムを作る」ということは，**「人間にとってより便利な道具を使う」**ことでもあります．ですから，ちょうどあなたが初めて自転車に乗るときに，何回も何回も転びながら，体で覚えていったこと．そして縦横に自転車を操縦できるようになったときの喜び．又，徒歩のときよりも飛躍的に行動範囲が広がったことの便利さ.

ピアノやエレクトーンでもそうですね.

美しい音色を出すための基本の繰り返し………

そうなのです．新たなものを学ぶための努力．この努力を忘れずに持ち続けていただければ，きっと高度なPC—9800シリーズの機能をあなた自身の手で引き出すことができるのです.

何度もいうようですが，本書では，

① 「16ビットパソコンをただおもちゃのようにいじくり回すだけではもったいない」

② 「16ビットパソコンを真に人間の役に立つ道具として使いたい」

この2点をバックボーンとして，自分で簡単なプログラムが作成できる水準に達すること，PC—9800シリーズの操作が行えることを学習の目標として編集いたしました.

本書を利用するにあたって

本書を書くに当って使用したPC—9800シリーズの機器は次の通りです.

PC-9801シリーズ本体及びキーボード
ディスプレイ装置
付属の取扱い説明書及びマニュアル

また，使用した言語はN_{88}—日本語BASIC(86)です.

なお，ディスプレイ装置は，白黒ディスプレイでもカラーディスプレイでも構いませんが，カラーグラフィック機能を理解していただくために，一応，高解像度カラーディスプレイ装置を対象にしています.

なお，本書においてはN_{88}—日本語BASIC(86)を起動するまでの操作に関しては説明をおこなっていません.

N_{88}—日本語BASIC(86)が起動され，キー入力ができる状態になった後の操作から説明が始まります.

N_{88}—日本語BASIC(86)の起動のしかたに関しては，本書とともに本体に添付されている「ガイドブック」を参照してください.

目　次

付　録

第1章　BASICプログラミングの基礎

本章のポイント

- ＢＡＳＩＣによるプログラムの作成と実行
- プログラムの修正方法
- プログラムの保存方法

第1節　BASICに触れる

会話事始め

まずはキーボードより次のようにキー入力してみて下さい．

[P][R][I][N][T][ ][1][+][2][↵]
スペースキー

[↵]（リターンキー）を押したとたんに，画面には次のように表示されましたね．

```
PRINT 1+2
 3 ←――――― 答がここに表示されます．
OK
```

同様に次のようにキー入力します．

```
PRINT 150-50
 100
OK
```

うまく画面に表示されましたか．もしうまく出来なかった人は
[P][R][I][N][T][ ][1][5][0][－][5][0] と入力したままになっていませんか．実はさきほどの例では最後に，リターンキー [↵] を押していましたね．もし押していない人は [↵] を押して下さい．

同じ画面になりましたね．

すなわち命令の最後には必ずリターンキー [↵] を押す必要があります．

以下，**本書では特別な指定がない限り，**

　PRINT　5＋4

という書き方をしますが，[↵] がないからといってリターンキーを押さなくてもよいのではありません．必ず押して下さい．

ダイレクトモードによるプログラムの作成

今，BASICの命令のひとつであるPRINT命令を使ってみました．キーボードからこの命令をキー入力すると，すぐに結果が出ました．

ちょうど電卓的な使い方ですが，このような使い方を**ダイレクトモード**（直接実行形）といい，BASICの特徴のひとつになっています．

これと対応し，もうひとつ**プログラムモード**（間接実行形）といわれる使い方があります．このプログラムモードによるプログラムの作り方は第2節で学習することとし，ここではダイレクトモードによるプログラム作りを学習しましょう．

PRINT命令に戻ります．次のようにキー入力します．

PRINT命令を用いて，次の問題を考えてみましょう．

例

田中君，伊藤君，石井君の身長はそれぞれ156cm，168cm，172cmです．3人の平均身長は幾らになるでしょうか．

```
PRINT (156+168+172)/3
 165.333
OK
```

（/ ← 除算記号として使います）

はじめての命令の使いごこちはいかがでしたか．

それでは命令の使用上のきまりについてまとめましょう．

● PRINT命令のまとめ

PRINT命令は，そのあとに続く式により計算を行い，その結果を直ちにディスプレイ画面に表示する命令です．

書き方

PRINT　式

・計算に使用できる記号には，次のものがあります．

加算（＋）……………＋（プラス）
減算（－）……………－（マイナス）
乗算（×）……………*（アスタリスク）
除算（÷）……………／（スラッシュ）
べき乗　……………∧（やまがた）
剰　余　……………MOD

（↑ BASICで用いる演算記号）

・また，PRINT命令の式の部分に"（クオーテーションマーク）で両方をくぎると，次のような文字の表示を行うことができます．

```
PRINT"OHAYO"
OHAYO
OK
```

（OHAYO ← "（クオーテーションマーク）で囲まれた中の文字）

・PRINT命令の簡略形として，?（疑問符）を用いることも可能です．

```
? 2*2
 4
OK
```

注意

通常，私達は乗算には「×」，除算には「÷」の記号を用いていますが，前者「×」は英文字のエックスと混同しやすいこと，後者の「÷」はディスプレイの表示できる文字にないことから，それぞれ「*（アスタリスク）」「／(スラッシュ)」を用いることにしています．

また，べき乗も，例えば4^5（4の5乗）と書きますが，これも表示できません．そのため4∧5と表現する訳です．

さて，複雑な計算になると，計算の途中結果を憶えておいて，その値を後で使うということはよくあります．

次の例はこのような問題をとり扱ったものです．

例

$a = (5+3)^2 \qquad b = \dfrac{(6-2)\times a}{a-10}$

を計算し，aの値とbの値を求めなさい．

```
LET A=(5+3)^2
OK                      ←―― 計算の答をAへ代入
LET B=(6-2)*A/(A-1Ø)    ←―― 計算の答をBへ代入
OK
PRINT A                 ←―― Aを表示せよ
 64                     ←―― Aの値
PRINT B                 ←―― Bを表示せよ
 4.74Ø74                ←―― Bの値
OK
```

● LET命令のまとめ

LET命令は，次に続く式の左辺の**変数**に右辺の計算結果を代入しなさい，という命令です．

書き方

LET　変数名＝式

なお，LETは省略が可能です．

注意

式の中の＝の記号は，数学でいう等号とは意味が違います．

BASICで使われる＝の記号は，「代入せよ」という意味です．

変数とは，値をメモリに記憶しておく場所であり，変数名とは，その場所につけられた名前です．

ですから，C＝C＋1の意味は「今までのCに1を加え，その値を新しいCの値にしなさい」ということです．（LETは省略されています）

これらを代入文とも呼んでいます．

ディスプレイ画面イメージを変えてみる

さて，次はプログラム作りとはチョット離れますが，皆さんが本体の電源スイッチをONにして以来，見つづけてきた画面があります．この画面イメージを変えてみませんか．

それではまず，今，写し出されている文字の色を覚えておいて下さい．

次に，次のようにキー入力してみて下さい．

```
CONSOLE ,,1,1
COLOR 4
OK
```

そして何か適当な文字をキー入力してみて下さい．

文字の色は緑色に変わりましたね．

この様に，COLOR命令に続いて数字を指定することにより画面の色を変えることができます．

書き方

COLOR
- 0……………黒
- 1……………青
- 2……………赤
- 3……………紫
- 4……………緑
- 5……………水色
- 6……………黄
- 7……………白

なお，CONSOLE命令については第6章で説明します．

それでは最初の色に戻しましょう．

```
COLOR 7
OK
```

次は画面上に表示する文字数の変更です．

さあ，キー入力してみて下さい．

```
WIDTH 80,25
OK
```

何か適当な文字を何行かキー入力してみて下さい．そして引き続き，次のようにキー入力します．

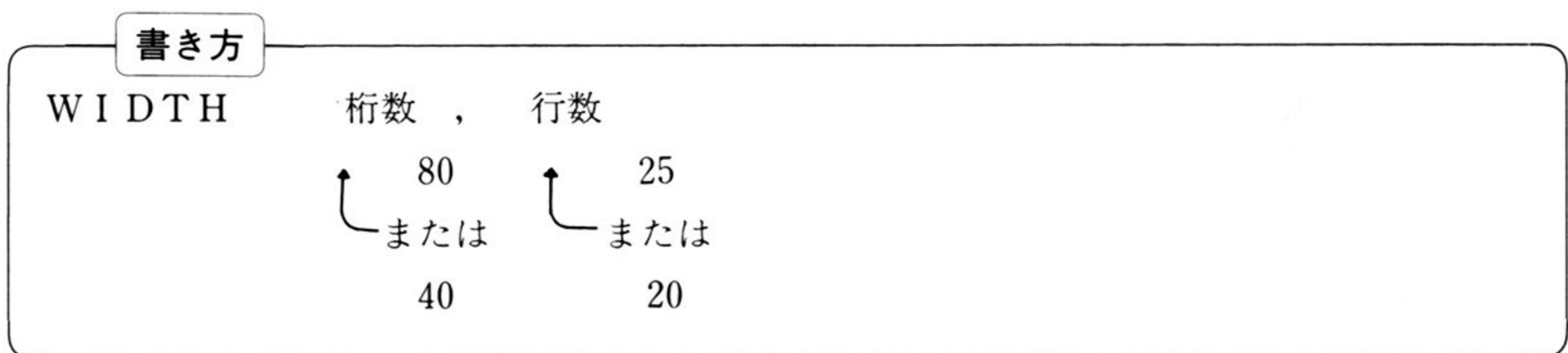
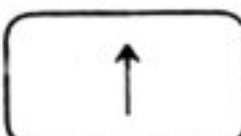
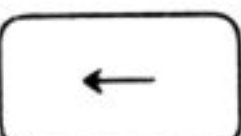
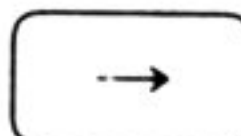
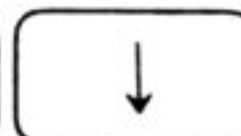
```
WIDTH 40,20
OK
```

引き続き，適当な文字をキー入力します．

どうでしたか．あなたはどちらの画面が気に入りましたか．

このように画面に表示できる文字数を変更できますので，あなたが見易いと思う画面表示が選べる訳ですね．

●WIDTH命令のまとめ

WIDTH命令は画面に表示する文字の行数と桁数を指定する命令です．

WIDTH命令で指定できる画面は4種類の組み合わせ方があります．あなたが見易いと思う画面にして下さい．

キーの押し間違いを直すには

さて，今まで何べんもキー操作を行ってきましたが，本書の例の通り，間違えずにキー入力することができましたか．それともピーというブザーの音とともにエラー・メッセージを表示されてしまいましたか．

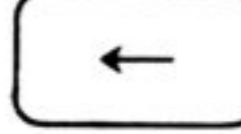
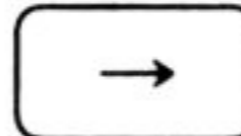
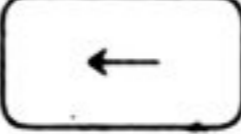
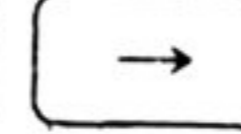
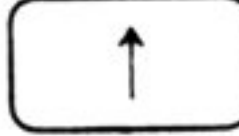
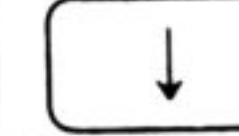
でも今までは，最初からキー入力をしなおすことにより何とかやってこれました．

しかしながら，一生懸命((カッコ)や”(クオーテーション)の記号を探し当て，何とかキー入力し続けながらも最後の1文字を間違えてしまったために，最初からやり直すのは何とも惜しい気持がしましたね．

そこでここでは，今キー入力している内容を生かし，間違い部分だけを修正する方法を学習しましょう．

[INS][DEL][↑][←][→][↓]は，誤り修正のために有効なキーなのです．

早速はじめましょう．

それでは，わざと間違えて，次のようにキー入力してみて下さい．

ただし [⏎] (リターンキー)はまだ押さないで下さい．もし既に押してしまった方はブザーのお世話になってしまいましたね．もう一度入れ直して下さい．

```
CALAR 4■
       ↑
       └── カーソル
```

カーソルは4のすぐ後ろで点滅してますね．

CALARとキーインしてしまった訳ですが，本当はCOLORと入力したかったとします．

修正に入りましょう．

① 先ず [←] を6回押し，先頭から2番目のAにカーソルを合わせます．

② そしてキーボードからOを1回押します．AがOに変り，そしてカーソルが次の文字Lの所にきました．

③ Lはそのままでいいですから [→] を1回押し，次の修正部分のAの位置にカーソルを合わせます．

④ AをOに直すため，Oを1回押します．AがOに変わり，カーソルはRのところにきています．

⑤ これで間違い部分はなくなりました． [⏎] を押してみましょう．

今の例では，同一行における修正でしたから [←][→] を使って修正を行った訳です． [↑][↓] は行を合わせるために用いることは分りますね．

次は [INS][DEL] の利用法です．

今，PRINT命令を押そうとして次のように押してしまいました．([⏎] はまだ押していません．)

```
PRIIN■
    ↑
    カーソル
```

さて，カーソルはNの直後で点滅しています．本当はPRINとキーインしたい訳ですね．
それでは修正しましょう．

① [←] を1回押し，Nにカーソルを合わせます．
② [DEL] キーを1回押します．Iがひとつ消えます．なお，カーソルは引き続きNの所に位置づけられ [DEL] による削除が可能な状態にあります．
③ PRINとなり，これで修正が終りましたので，カーソルをもとの位置に戻すため [→] を1回押し，Nの次にカーソルを位置づけます．
④ 引き続き，以降のキー入力を行います．

```
PRINT 5+3
 8
OK
```

次は [INS] です．PRINT5＋3を押そうとして，次のように押してしまいました．

```
PRIT 5+3■
        ↑
        カーソル
```

それでは修正しましょう．

① [←] を5回押し，カーソルをTの所に位置づけます．
② [INS] を1回押します．(画面変化はありません)
③ 挿入したい文字Nを1回押します．IとTの間にNが入りましたね．

なお，カーソルは引き続きTの所に位置づけられ， [INS] による挿入が可能な状態にあります．ですから引き続き文字を挿入することができます．挿入の終了はカーソルの位置を動かすことにより行われます．

第２節　プログラムの作成と実行

第１節では，BASICの手始めとして，電卓的な使い方——**ダイレクトモード**——による命令の実行を学習してきました．

このダイレクトモードでは，キー入力した命令をその場ですぐに実行して，即座に結果を得ることができました．

このダイレクトモードで実行することは，すぐ結果が得られるメリットがありますが，その反面，同じようなことを何度も実行させたい場合は，その都度，キー入力しなければならず，不便さを感じますね．

また，このような使い方では，人間のキー入力速度に左右され，コンピュータ本来の機能を発揮させることができません．コンピュータがこれほどまでに発展してきた理由は，人間にとって便利な機械だったからです．

コンピュータの超能力発揮の秘密でもあった，仕事の手順を指示したプログラムをいったん覚えさせさえすれば，あとはコンピュータがこのプログラムに従って，速く正確に何度でも仕事を行うことができるのでしたね．

本章では，コンピュータのもうひとつの使い方でもある**プログラムモード**(間接実行形)でのプログラムの作り方について学習します．

ことば

ダイレクトモード

これは，コンピュータが，１個の命令を受け取ると，ただちにそれを実行する形式です．その場かぎりの計算や，記憶する必要のないものは，ダイレクトモードで実行させます．ダイレクトモードでは，１行ずつ実行していくので，プログラムはメモリに残りません．

プログラムモード

このモードでは，すぐに命令を実行せず，メモリに記憶します．ですから，複数個の命令が集まって，１つの処理手順が作られることになります．この命令の集まりが「プログラム」です．

プログラムは，実行が終わっても，メモリに消えずに残っているので，何回でも実行できます．この点がダイレクトモードと異なります．

プログラムを作る

BASICの命令を，その場で即座に実行するのではなく，いったんプログラムとして記憶するプログラムモードにするには，どのようにすればよいのでしょうか．

今迄学習してきた命令の扱いに対し，単に命令の前に番号を付けて入力すればよいのです．それでは早速やってみましょう．

例

$a=(5+3)^2$　　　$b=\dfrac{(6-2)\times a}{a-10}$

を計算し，aの値とbの値を求めなさい

この問題は，第1節で既にダイレクトモードで実行していますね．
プログラムモードでは次のようになります．

```
NEW
10 A=(5+3)^2
20 B=(6-2)*A/(A-10)
30 PRINT A
40 PRINT B
50 END

RUN
 64
 4.74074
OK
```

コンピュータに覚えこませた手順（プログラム）：NEW〜50 END
← 実行指示（RUN）
← Aの値（64）
← Bの値（4.74074）

このプログラム例は，第1節の問題と同じですが，プログラムモードにするため，まず**行番号**といわれる番号を付けた5行の命令をコンピュータに覚えさせます．そして，プログラムの実行はRUN命令をキー入力することにより順次命令が実行され，最後にA，Bの値が表示され，プログラムの実行は終了します．

書き方

行番号　　　命令文

・BASICの各プログラム行は，行番号で始まらなければなりません．
・行番号は1から65529までの整数です．
・行番号は，プログラム行をメモリに格納する順序を示し，実行も行番号の若い方から行われます．

● **NEW命令のまとめ**

メモリ上のプログラムを消去します．

書き方

NEW

● END命令のまとめ

プログラムの実行を終了させます.

書き方

```
END
```

● RUN命令のまとめ

メモリ上にあるプログラムの実行を開始させます.

書き方

```
RUN　　行番号
```

・行番号を指定すると，その行から実行が始まります.

・行番号が省略されたときは，一番小さい行番号から実行を始めます.

プログラムの表示

今，実行したプログラムは，メモリ内に残っている限り，何度でも実行することができます.試してみましょう。

まず，メモリにプログラムが記憶されているか確認してみましょう.

現在入力されているプログラムを，画面に表示させるにはLIST命令を使います.

```
LIST
10 A=(5+3)^2
20 B=(6-2)*A/(A-10)
30 PRINT A
40 PRINT B
50 END
OK
```

引き続き，RUNとキー入力して下さい.

```
RUN
 64
 4.74074
OK
```

同じ結果が得られました.

● LIST命令のまとめ

書き方

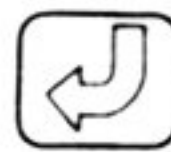
LIST　開始行番号ー終了行番号

・ 行番号が省略された場合は，プログラムすべてを画面表示します．

実行時にデータを与えるプログラム

今迄取扱ってきたプログラムは，計算データが予めプログラムの中に組み込まれた形式のものでした．

ここでは，実行時に計算データを与えることにより，繰り返し実行できる形式のプログラムを学習しましょう。

例

身長から求めるあなたの標準体重は何kgでしょうか．

男性の標準体重kg＝身長cm－110cm

早速キー入力してみましょう．（行番号60のEND⏎を入力したあとでSTOPを押して下さい．）

```
NEW
AUTO
10 INPUT "ナマエ";N$
20 INPUT "シンチョウ";T
30 R=T-110
40 PRINT "ナマエ=";N$
50 PRINT "リソウタイジュウ=";R
60 END
70
OK
```

プログラムが完成しました．それではあなたの標準体重を求めてみましょう．

実行させるためには…… RUN ⏎ でしたね．

画面にはナマエ？が表示され，カーソルが点滅しています．これはINPUT命令を実行して，あなたの名前をキーボードから入力してくるのを待っている状態です．

あなたの名前をキー入力してみて下さい．

名前は英字でもカナ文字でもかまいません．

名前をキー入力すると，即座に次のINPUT命令による入力待ちとなります．

```
RUN
ナマエ? サトウイチロウ
シンチョウ? ■
```

カーソル

身長をキー入力したとたん，最終的に求める答が即座に表示されました．

```
RUN
ナマエ？ サトウイチロウ
シンチョウ？ 167
ナマエ=サトウイチロウ
リソウタイシ゛ュウ= 57
OK
```

これで，ひとり分の理想体重の計算ができました．繰り返し計算が可能ということは，引き続きRUN状態に戻るということです．

試しにＲＵＮ命令をキー入力して下さい．

```
RUN
ナマエ？
```

確かに次の人の計算ができそうですね．続けて計算をしてみて下さい．

● AUTO命令のまとめ

行の先頭に自動的に行番号を付加します．

書き方

AUTO　行番号，増分

・行番号を最初の番号とし，増分の値を増やしながら行番号をつけていきます．

・行番号，増分とも省略されたときは両方とも10がとられます．

● INPUT命令のまとめ

キーボードから入力されるデータを変数に入れます．

書き方

INPUT　〔"文字列"；〕変数名

(注)〔　〕内は省略可

・INPUT文が実行されると，画面に入力を要求する「？」が表示されます．

　これに続いてキー入力されたデータを変数名の値とします．

・文字列を指定すると，「？」の前に文字列がそのまま表示されます．

・入力するデータの型（数値，文字など）と変数の型は一致していなければなりません．

ことば

変数の種類（型）… 変数値として数字や文字データを扱うことができます．また，種類も次に示す4種類があり，どの種類であるかは変数名の最後に付けた文字で区別します．

整数型……………%

単精度実数型……！

倍精度実数型……#

文字型……………$

（詳細についてはリファレンス・マニュアルを参照して下さい）

● PRINT命令の補足

第1節でもPRINT命令について説明しましたが，ここでは追加された機能について説明します．

ことば

PRINT　式；式………

・式を複数個書くことができます．その場合は，式と式の間を；（セミコロン）で区切ります。

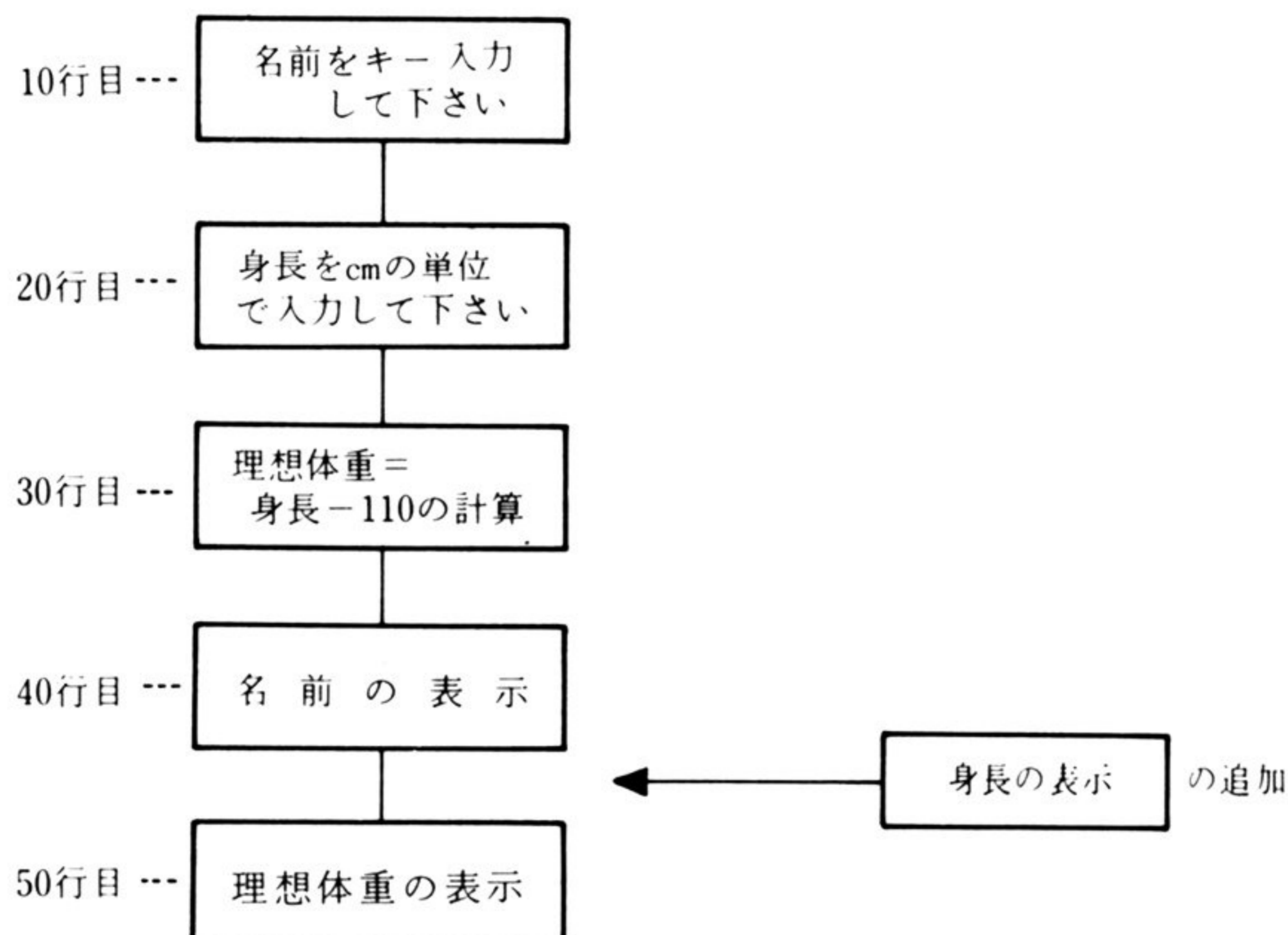
第3節　プログラムの修正

さて，今，作成した理想体重を求めるプログラムを再度見直してみますと，画面に表示されている結果は，名前と理想体重だけです．

折角，自分の身長をキー入力した訳ですから，それも表示したいですね．

それでは，さきほどのプログラムに身長も表示できるように変更してみましょう．

① まずLIST命令で画面にプログラムを表示します．

このプログラムは次の図に示す処理手順ででき上がっていました．

② 次に修正場所を考えます．

身長の表示を追加する場所は，名前の表示の次あたりでも構いませんね．

③ 追加したい項目をＢＡＳＩＣの命令におきかえます．

PRINT “シンチョウ＝”；T

④ 元となるプログラムに追加します．

今，追加したい場所は40行と50行の間ですから，この間に挿入できる行番号を見つけ出し，次のようにキー入力します．

```
45 PRINT "ｼﾝﾁｮｳ=";T
```

追加がうまくなされているかどうかを確認するために，再度LIST命令をキー入力します．

```
LIST
10 INPUT "ﾅﾏｴ";N$
20 INPUT "ｼﾝﾁｮｳ";T
30 R=T-110
40 PRINT "ﾅﾏｴ=";N$
45 PRINT "ｼﾝﾁｮｳ=";T
50 PRINT "ﾘｿｳﾀｲｼﾞｭｳ=";R
60 END
RUN
```

試しに，RUN命令を実行してみて下さい．

実行結果も上々です．

再度プログラムを見ますと，あとからプログラムを修正したことを行番号が物語っていますね．

また，何回もプログラムを修正することによって挿入できる行番号がなくなってしまう等の不都合がでてきます．

そのため，行番号をきれいな行番号にふり直すRENUM命令という便利な命令が用意されています．

書き方

RENUM

実際にキー入力してみて下さい．

プログラムの修正の仕方

プログラムの修正の仕方についてまとめてみましょう．

1 現在ある行の内容を変更したい時は，同じ行番号を使って新しい行を入力します．

例えば，女性の理想体重は身長－105ですから，もしさきほどのプログラムを女性の場合のプログラムに変更する場合は，次のようにキー入力することになります．

```
30 R=T-105
```

2 プログラムに，新しく命令文を追加したい時は，追加したい行間の２つの行番号の間の数値を新しい行番号にすればよいのです．

さきほどの45行目の追加方法がこれに当ります．

3 不要になった命令を削除する場合には，次の２つの方法があります．

① 削除したい行があれば，行番号だけをキー入力して ⏎ を押します．

② 何行にもわたって削除する場合は，DELETE命令を使います．

書き方

DELETE　開始行番号－終了行番号

第4節　プログラムの保存

さて，今，一生懸命になって作っているプログラムは，メモリ上に記憶されている訳です．ところでメモリ上のプログラムは電気的に記憶されていますので本体の電源を「OFF」にすると，プログラムは跡かたもなく消えてしまいます．

また，NEW命令を実行しても同様です．

そのため，必要に応じて繰返し実行するようなプログラムは，別の場所へ保存しておく必要があります．

代表的なプログラムの保存方法について説明します．

1　プログラムをプリンタ用紙に印字する．

メモリ内にあるプログラムを，全部または一部をプリンタを通して用紙に印字することにより，保存しようというものです．

書き方

LLIST　開始行番号—終了行番号

・印字内容はLIST命令で画面表示したものと全く同じものです．

2　プログラムをフロッピィディスクに保存（**セーブ**）する．

メモリ内にあるプログラムを，フロッピィディスク装置を通してフロッピィディスク媒体に保存しておこうというものです．

この保存を一般にはセーブと呼んでいます．

書き方

SAVE　“ドライブ番号：ファイル名”

・ドライブ番号はフロッピィ媒体の入っている装置の番号で，ファイル名はプログラムにつける名前です．

・ファイル名は9文字以内です．

例えば，今，メモリ上に記憶されているプログラムを1番のドライブに入っているフロッピィにセーブしたい．また，名前をREIDAIとしたときのキー入力は次のようにすればよいのです．

```
SAVE "1:REIDAI"
```

メモリへの格納

一方，セーブしているプログラムをメモリに格納する場合は，LOAD命令を用います．

書き方

LOAD　“ドライブ番号：ファイル名”

・ファイル名はセーブの時につけた名前です．

さきほどセーブしておいたプログラム“REIDAI”をロードする場合のキー入力は次のようになります．

```
LOAD "1:REIDAI"
```

ところで，今迄本体に何らかの動作を起こさせる指示を「命令」と呼んできましたが，実際には「文」と「コマンド」とに区別されます．

・文……………プログラムの中で用いられる命令です．今迄でてきたものでは，“PRINT”，“LET”，“CONSOLE”，“COLOR”，“WIDTH”，“END”等がこれにあたります．

・コマンド……プログラム全体あるいはある部分を単位として処理を行う命令です．今迄でできたものでは，“NEW”，“RUN”，“LIST”，“AUTO”，“RENUM”，“DELETE”，“LLIST”，“SAVE”，“LOAD”等がこれにあたります．

第2章　計算・印字プログラムの作成

第1章では，操作法を中心にプログラムの作り方等について学習しました．本章からは，実際のプログラム例を通して，プログラム作成の基本テクニックを学びましょう．

本章のポイント

- ∘ データ集計の仕方
- ∘ プリンタへの処理結果の印字の仕方
- ∘ 処理を繰り返して行う方法

第1節　データの集計と一覧表の作成

この節では，試験の点数表を作成するプログラム例で，データの集計と一覧表の作成方法を学びます．

1　試験結果一覧表の作成（例題2－1）

例題の概要

これから考えていただく例題は，次のようなデータを基に，試験結果一覧表を作成するプログラムです．

試験採点結果データ（入力データ）

受験番号	氏　　　名	国語得点	数学得点	英語得点
101	アオヤマ　タロウ	85	70	80
102	アライ　イチロウ	75	80	90
103	イワタ　ユキオ	70	75	85
104	ウエダ　マサオ	100	95	70
105	エトウ　ヤスヒコ	90	80	65
106	オオタ　カツオ	80	75	100
107	カトウ　タカシ	90	90	90
108	キトウ　ミチオ	75	60	65
109	サトウ　アキヨシ	80	80	100
110	シミズ　シンイチ	100	100	100

◎　試験結果一覧表の印字形式

用紙の縦位置を表す数字（行）

（注）X～Xで示されている位置はデータを印字する位置です．

処理の条件

・国語，数学，英語はそれぞれ100点満点とします．

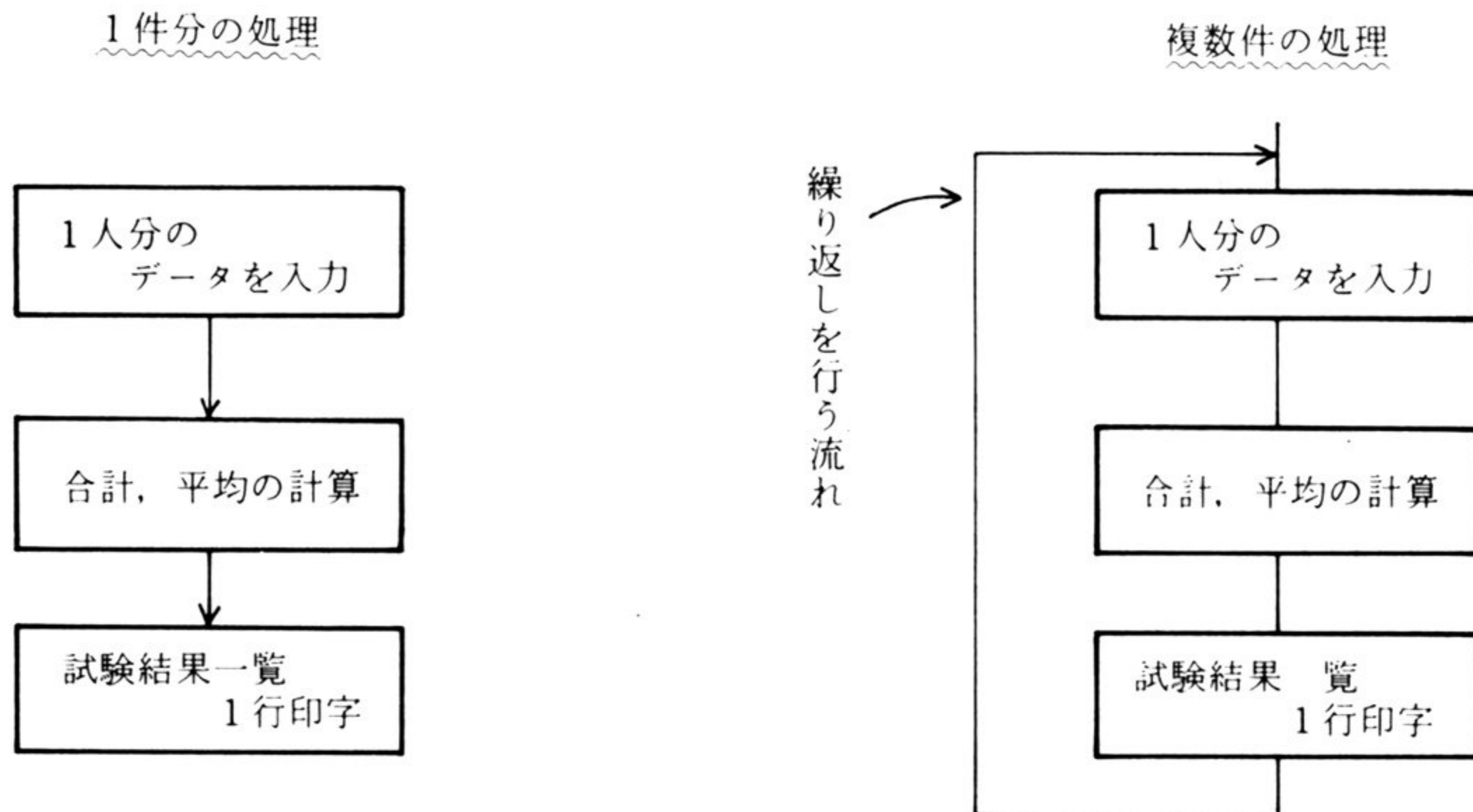
・各人のデータは，1行に印字し，各印字行の間隔は1行あけます．

・平均点は，合計÷3で計算します．小数部は切り捨て，整数部のみを印字します．

・全員（10人）の得点データを印字したら，プログラムを終了します．

プログラム作成上の注意点

プログラムでは，10人の得点データを試験結果一覧として印字します．

1件分の処理は，受験番号，氏名，国語得点，数学得点，英語得点を入力し，合計，平均を計算し，それらを1行に印字します．そして，これを10回繰り返すと一覧表ができます．プログラムを考えるときは，1件分（この場合は1人分）の処理手順だけを考えて，それを10件，繰り返し行うようにします．処理手順を図で表すと次のようになります．

BASICでは，繰り返しを指示する命令として，GOTO，FOR～NEXTなどの命令が用意されています．

2 プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．

```
10 DATA 101,ｱｵﾔﾏ ﾀﾛｳ,85,70,80,102,ｱﾗｲ ｲﾁﾛｳ,75,80,90
20 DATA 103,ｲﾜﾀ ｺｷｵ,70,75,85,104,ｳｴﾀﾞ ﾏｻｵ,100,95,70
30 DATA 105,ｴﾄｳ ﾔｽﾋｺ,90,80,65,106,ｵｵﾀ ｶﾂｵ,80,75,100
40 DATA 107,ｶﾄｳ ﾀｶｼ,90,90,90,108,ｷﾄｳ ﾐﾂｵ,75,60,65
50 DATA 109,ｻﾄｳ ｱｷﾖｼ,80,80,100,110,ｼﾐｽﾞ ｼﾝｲﾁ,100,100,100
60 LPRINT TAB(27);"ｼ ｹ ﾝ ｹ ｯ ｶ   ｲ ﾁ ﾗ ﾝ":LPRINT
70 LPRINT "      ﾊﾞﾝｺﾞｳ        ｼ    ﾒ     ｲ           ｺｸｺﾞ";
80 LPRINT TAB(45);"ｽｳｶﾞｸ     ｴｲｺﾞ   ｺﾞｳｹｲ    ﾍｲｷﾝ"
90 LPRINT
100 FOR I=1 TO 10
110 READ BANGO,SHIMEI$,TEN1,TEN2,TEN3
120 LPRINT TAB(6);BANGO;TAB(17);SHIMEI$;TAB(37);TEN1;TAB(47);TEN2;TAB(54);TEN3
;
130 LPRINT TAB(62);TEN1+TEN2+TEN3;TAB(70);INT((TEN1+TEN2+TEN3)/3)
140 LPRINT
150 NEXT I
160 END
```

それでは，プログラムを分けて見ていきましょう．

①

```
10 DATA 101,ｱｵﾔﾏ ﾀﾛｳ,85,70,80,102,ｱﾗｲ ｲﾁﾛｳ,75,80,90
20 DATA 103,ｲﾜﾀ ﾕｷｵ,70,75,85,104,ｳｴﾀﾞ ﾏｻｵ,100,95,70
30 DATA 105,ｴﾄｳ ﾔｽﾋｺ,90,80,65,106,ｵｵﾀ ｶﾂｵ,80,75,100
40 DATA 107,ｶﾄｳ ﾀｶｼ,90,90,90,108,ｷﾄｳ ﾐﾁｵ,75,60,65
50 DATA 109,ｻﾄｳ ｱｷﾖｼ,80,80,100,110,ｼﾐｽﾞ ｼﾝｲﾁ,100,100,100
```

各行の最初の「DATA（データ）」は，データを定義するための命令です．

ここでは，試験データ（入力データ）が定義されています．10人分の受験番号，氏名，国語得点，数学得点，英語得点です．

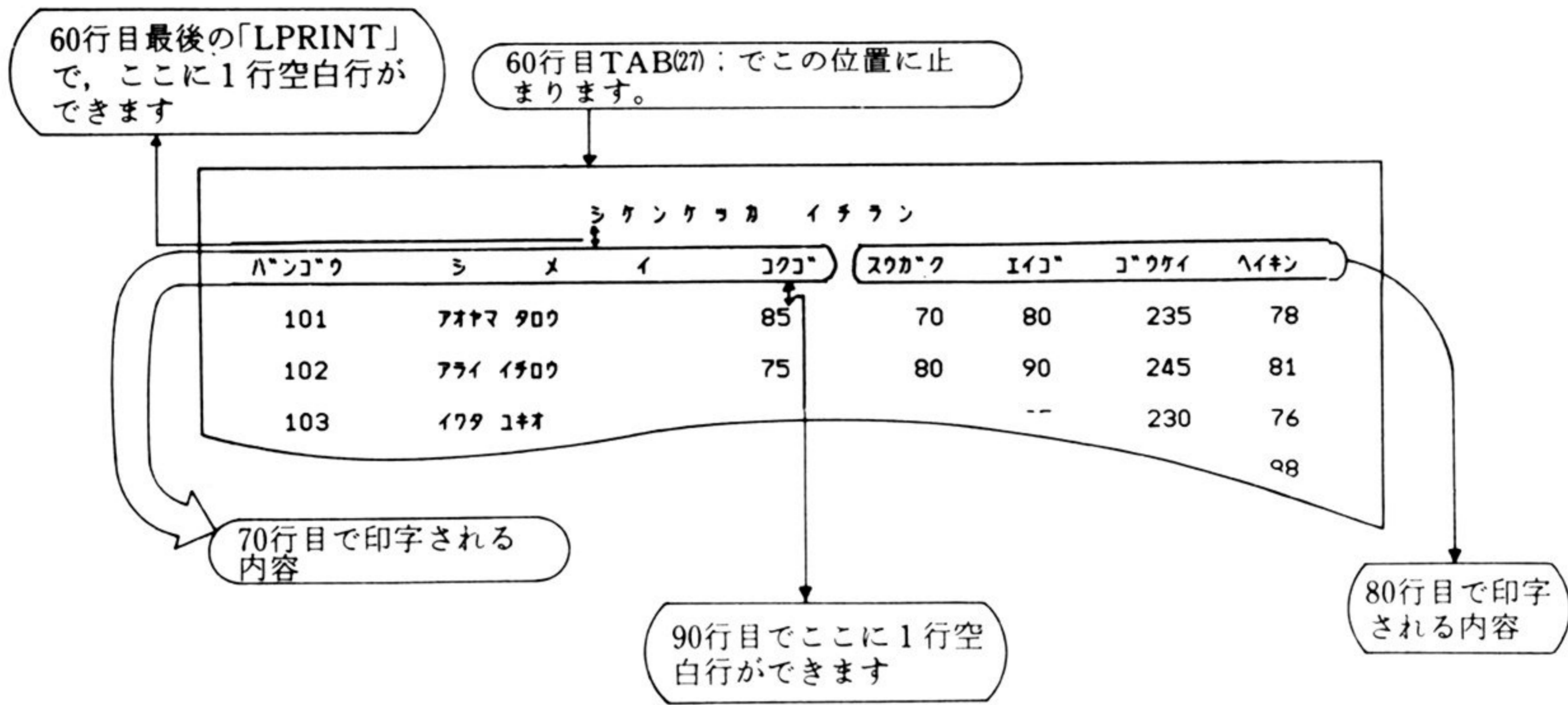
②

```
60 LPRINT TAB(27);"ｼ ｹ ﾝ ｹ ｯ ｶ   ｲ ﾁ ﾗ ﾝ":LPRINT
70 LPRINT "     ﾊﾞﾝｺﾞｳ        ｼ    ﾒ     ｲ          ｺｸｺﾞ";
80 LPRINT TAB(45);"ｽｳｶﾞｸ     ｴｲｺﾞ   ｺﾞｳｹｲ    ﾍｲｷﾝ"
90 LPRINT
```

各行の最初の「LPRINT（エルプリント）」は，見出し，入力データ，処理結果などをプリンタに印字させる命令です．ここでは，一覧表の見出しや項目の見出しを印字しています．

60行と80行の「LPRINT」で指定されている「TAB（タブ）(nn)」は，印字の開始位置を指定します．この場合は，プリンタ用紙の現在行の左端から数え，最初の文字を印字するまでの間隔数となります．本テキストでは，プリンタ用紙の左端（1桁目）を0桁目として数えていくことにします．

つまり60行では，27桁目に位置づけ，27桁目から引用符（″）で囲まれた「シ△ケ△ン△ケ△ッ△カ△△△イ△チ△ラ△ン」を印字することになります．

60行目の右端にある「LPRINT」と，90行目の「LPRINT」は改行だけを行います．

注意

①「TAB（nn）」で，左端からの桁位置を指定することにより，その桁の指定桁位置に位置づけることができますが，画面，プリンタともその行の左端を「0」から数え始める決まりとなっています．

②「LPRINT」を実行すると，データを印字後，自動的に改行します．しかし，命令の最後に「；」や「，」が指定されていると改行せずに，最後の印字文字の次の桁に位置づけられたままになっています．

（注）「△」は空白1桁を表わす記号です．

③

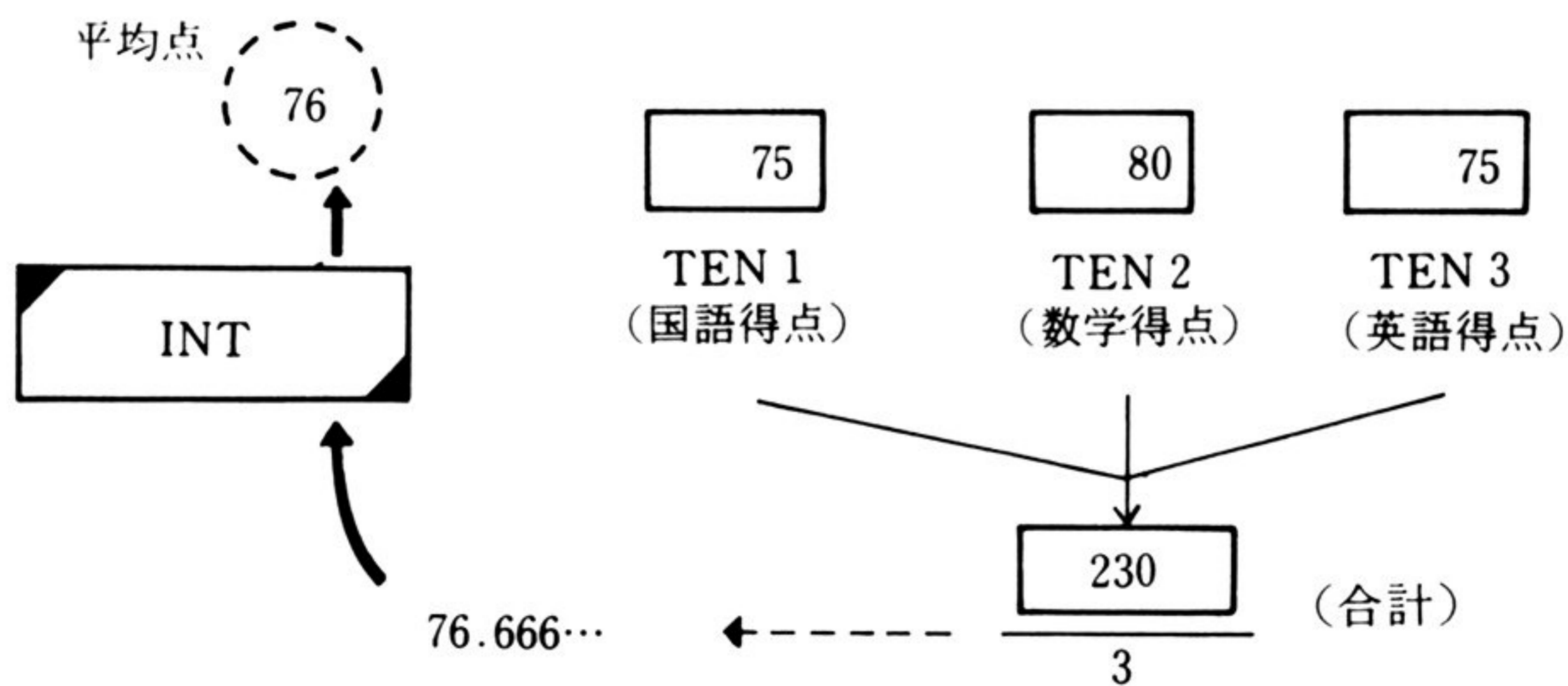
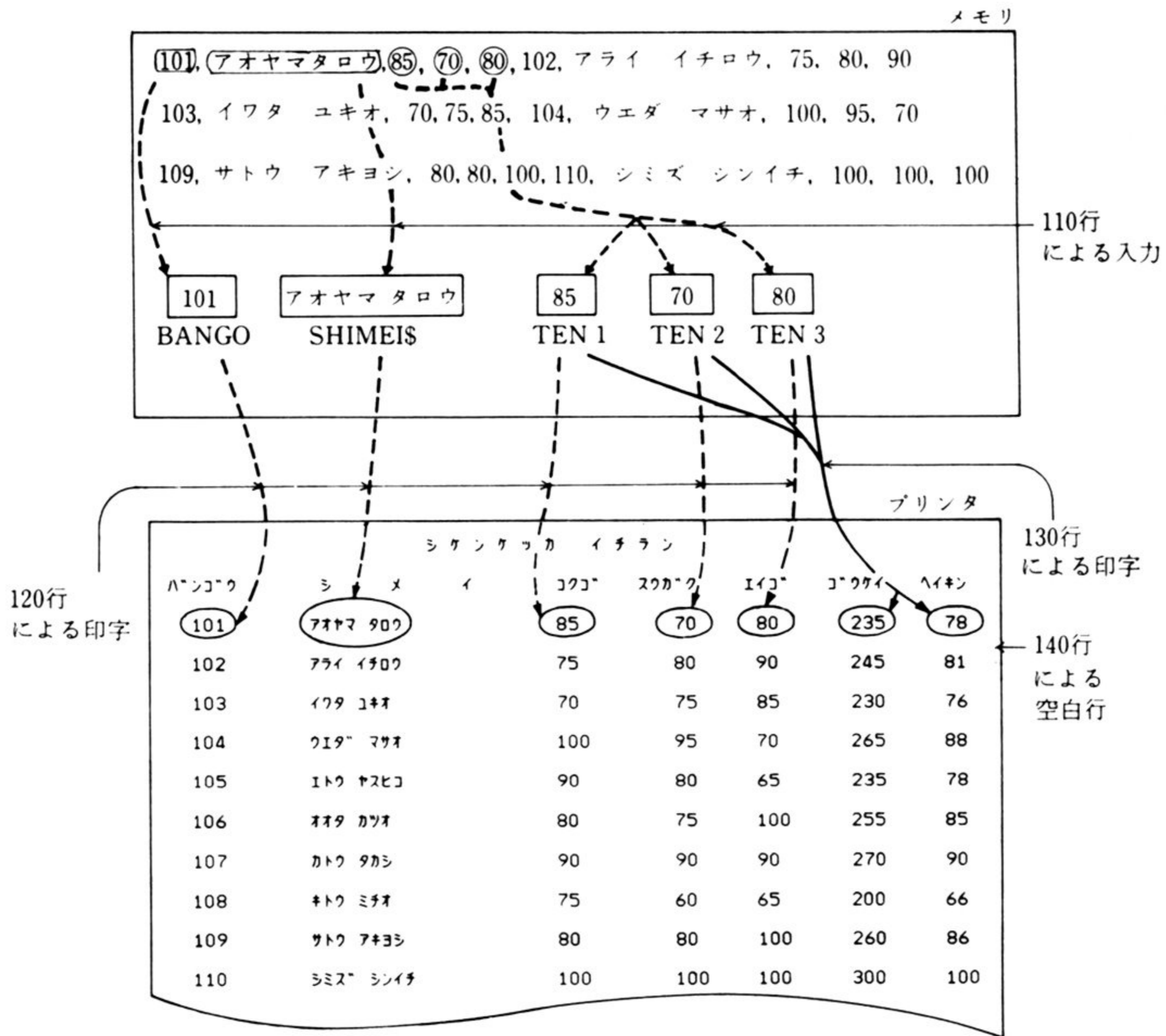
```
100 FOR I=1 TO 10
110 READ BANGO,SHIMEI$,TEN1,TEN2,TEN3
120 LPRINT TAB(6);BANGO;TAB(17);SHIMEI$;TAB(37);TEN1;TAB(47);TEN2;TAB(54);TEN3;
130 LPRINT TAB(62);TEN1+TEN2+TEN3;TAB(70);INT((TEN1+TEN2+TEN3)/3)
140 LPRINT
150 NEXT I
160 END
```

100～160行は，10人分の試験結果を印字します．110行から140行が試験結果を１件分入力し印字するための処理で，それを10件分繰り返す（10人分の印字を行う）命令が100行の「FOR（フォー）」と150行の「NEXT（ネクスト）」です．

100行で「Ｉ」（変数）の初期値として１を入れ，150行の「NEXT」で「Ｉ」の値を１ふやします．110～140行の処理を10回繰り返し，「Ｉ」が10を超えた，つまり11になったところで，この処理は終ります．

つまり，この例では，110行から140行までが，１から10まで10回繰り返されることになります．

１人分の処理としては，110行の「READ（リード）」命令で，「BANGO」，「SHIMEI＄」，「TEN1」，「TEN2」，「TEN3」という変数に10行～50行で定義されている試験データを読み込み（記憶し），120行で所定の位置に印字します．130行では，３科目の合計と平均点を印字します．140行は行間を1行あける（空白行）のための命令です．

ここで平均点を求める計算式の前に「INT（インテジャー）」という指定がありますが，これは平均点として実数（小数部をもつ値）がでた場合に，整数部分だけを取り出す命令です．（処理の条件として平均点は整数部だけを印字することになっています）

注意

110行の「READ」文が再度実行されると，DATA文で定義されている次のデータ（２回目の実行の時は「102」，「アライ　イチロウ」～「90」）が続けて入力されてきます．

つまり，ＤＡＴＡ文で定義されているデータは，ＲＥＡＤ文で少しずつ取り出しながら利用していくわけです．

③ 処理の手順

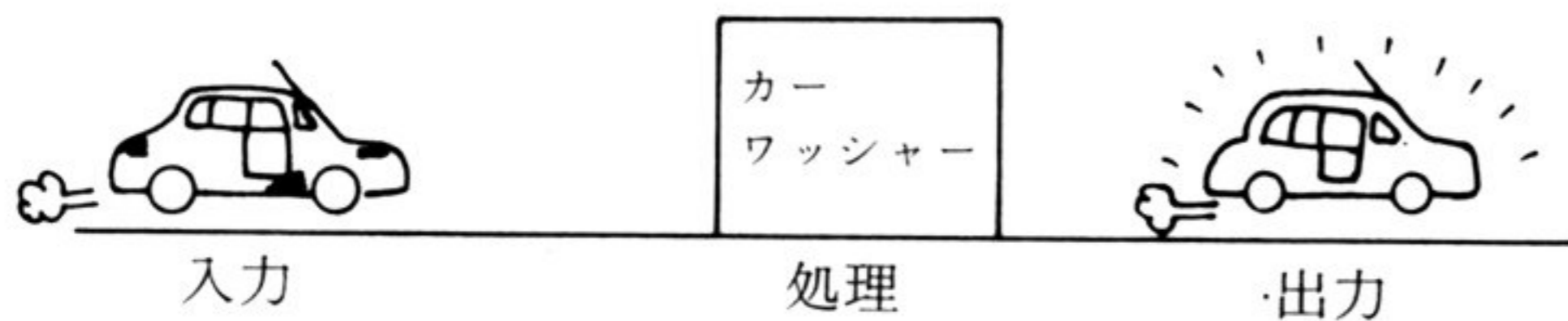
②で取り扱ったプログラム例の実行手順を，もう一度まとめてみましょう．

ここでは，プログラムを図を使って表わしてみます．

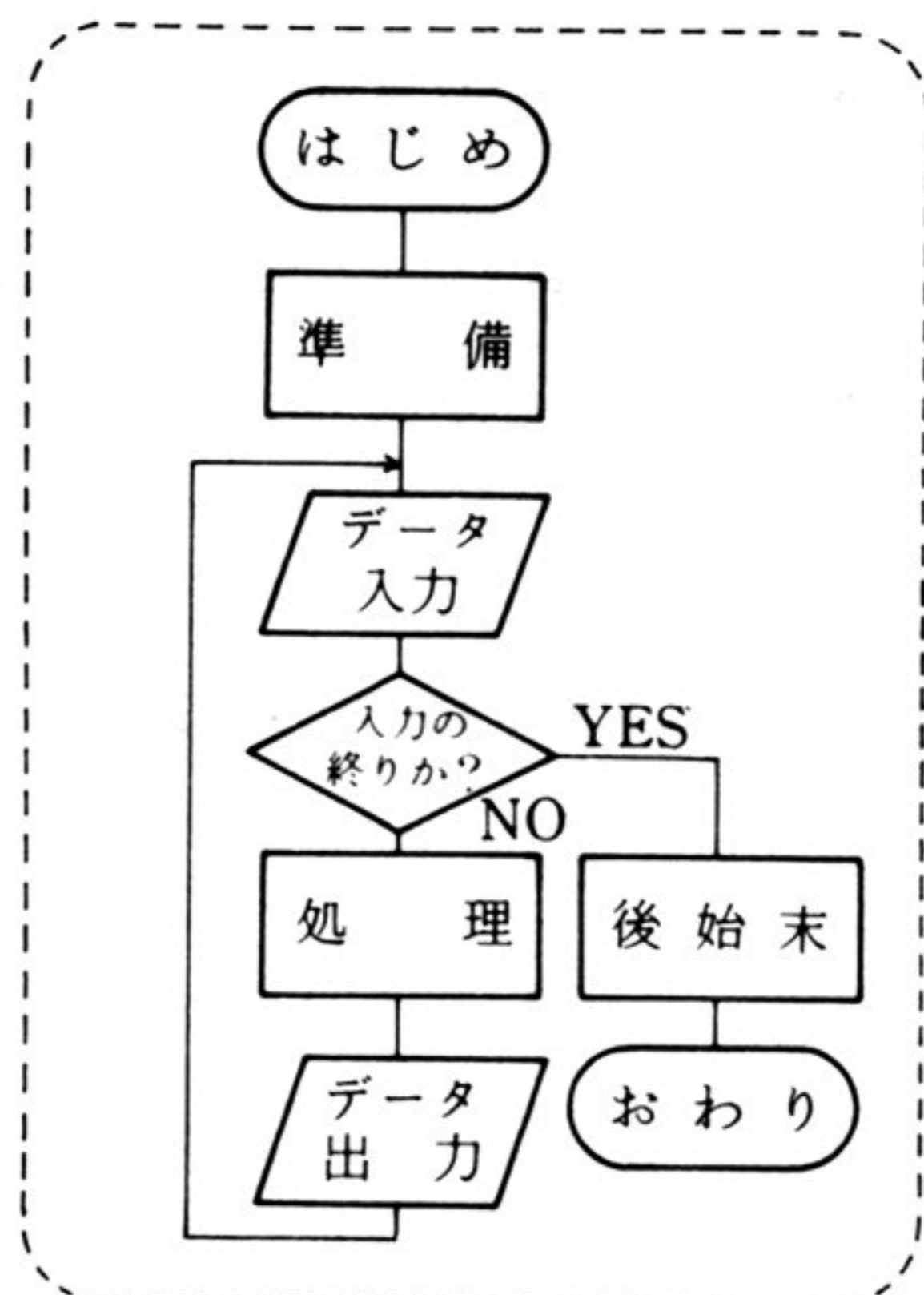
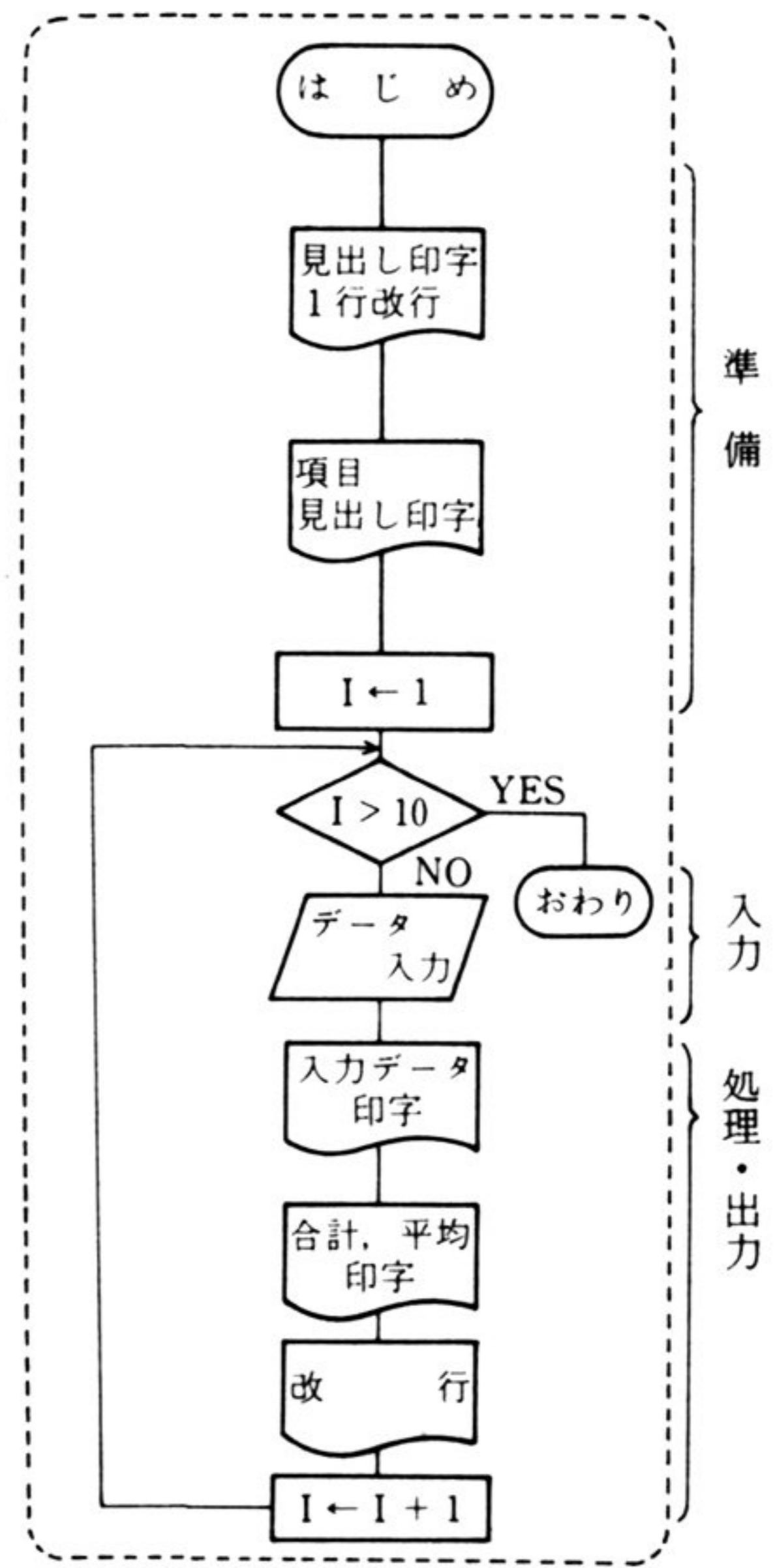
右図をご覧下さい．これは，プログラム例を処理ごとに分けて図で表わしたものです．このように，プログラムの実行の流れを，図（記号）を使って表わしたものを「フローチャート（流れ図）」と呼びます．

処理の手順としては，まず「大見出し」，「項目見出し」を印字しておきます．そして，１件分（例では１人分）の処理を実行します．１件分の処理は，データを入力後，そのデータに対して処理を行い（計算等），結果を出力（プリンタに印字）しています．これを何回も繰り返せるように指示することによって，多数のデータを同様に処理していくことができるわけです．

ある処理結果が欲しい時には，その元になる入力データが必要ですし，その入力データに対して必要な処理を行って処理結果を導き出せるわけです．この手順は，このプログラムだけでなく，皆さんがこれから作成する多くのプログラムの基本型になります．

洗車の過程

プログラムの基本型を流れ図で表わすと右のようになります．

本章の例では取り扱っていない部分もありますが，繰り返し処理を行っているプログラムは，ほとんど右図のような形になります．

各処理においては実際に，どのようなことを行うのでしょうか．以下にいくつかの例を挙げておきます．

準　備………・ファイル準備（フロッピィディスクのデータを入出力する際必要）
　　　　　…・見出しの表示，印字，初期画面の表示

データ………・DATA文で定義したデータをREAD文で入力
入　力　　…・フロッピィディスクに登録したデータを入力（次章で扱います）
　　　　　…・キーボードからのデータ入力

処　理………・計算，判断（比較），移送（データを他の変数に移す）

データ………・プリンタへの印字
出　力　　…・画面への表示
　　　　　…・フロッピィディスクへの登録（書き出し）

後始末………・ファイル後始末（ファイル準備に対応する作業）
　　　　　　・合計処理，画面の初期化など

プログラムを終わりにするかどうかの判断は，各プログラムの処理内容により変わってきますが，基本的には入力してくるデータがなくなった時にプログラムを終了します．

終了かどうかの判断を行うタイミングは，データ入力を行った後に行う場合が多くなるでしょう．

4 命令の説明

この章で新しく出てきた命令を見ていくことにしましょう．

①DATA文

READ文で，ある変数に入力するデータを定義します．

書き方

```
DATA　　定数〔，定数〕…
```

DATA文を書く時の注意は，次のとおりです．

・定数はコンマ（,）で区切りながら1行の範囲（255文字以内）で指定できます.
・READ文でデータを読み込む時には，DATA文の最初のデータから順番に入力し，再度READ文を実行すると，最後に読み込まれた次のデータから入力されていきます.
・プログラム中には，DATAが何行あってもかまいません．各データは，プログラム中で連続しているデータとして扱われます.

②READ文

DATA文で定義したデータを，変数名に読み込みます.

書き方

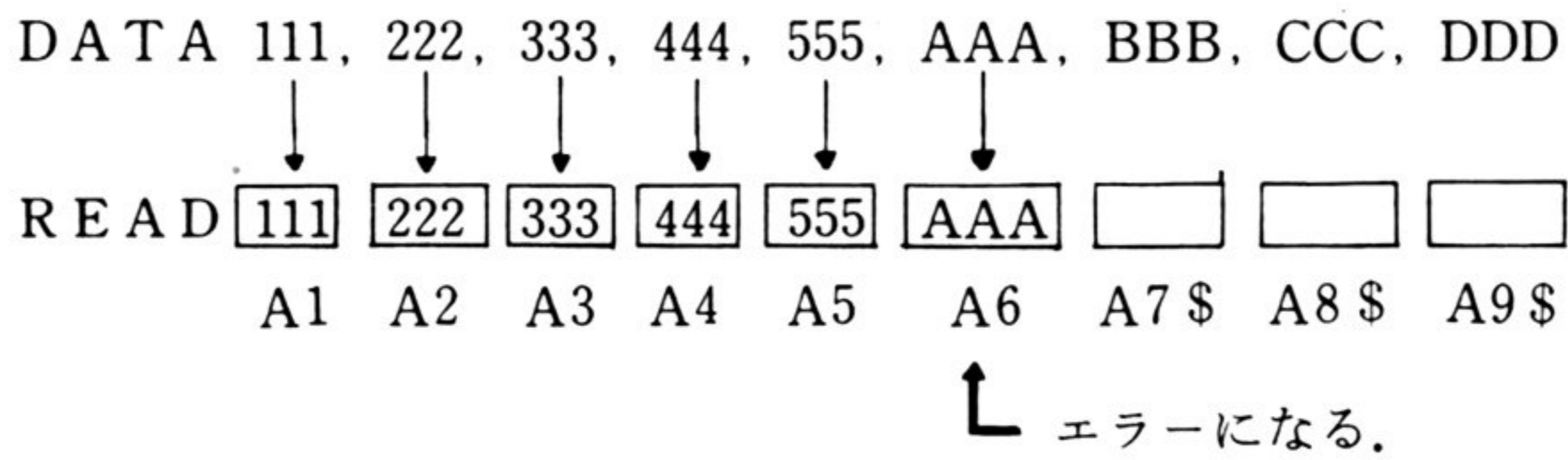
READ　変数名〔，変数名〕…

READ文を書く時の注意は，次のとおりです.

・READ文で入力してくるデータと，それを入れる変数の型（数値，文字）は一致していなければなりません.
・変数名はコンマ（,）で区切って複数個指定できます.

例

```
10 DATA 111, 222, 333, 444, 555
20 DATA AAA, BBB, CCC
30 READ A1, A2, A3, A4
40 READ A5, A6
50 READ A7$, A8$, A9$
60 DATA DDD
         ～
```

左のようなプログラムの場合，10行，20行，60行のデータはプログラム中で継続していることになります．つまり，30行～50行のREAD文の実行により，各変数には次のようにデータが読み込まれます．

この例では，A6という変数にデータを読み込む際にエラーになります．その理由は，A6という変数が数値だけを扱える数値変数なのにもかかわらず，「AAA」という文字を入力してしまったからです．

③LPRINT文

データをプリンタに印字します．

書き方

LPRINT 〔式{ , ; }式〕…〕

LPRINT文を書く時の注意は，PRINT文と同じです．

④TAB関数

データの画面表示や印字を行う場合，その行の特定の桁位置に位置づけします．

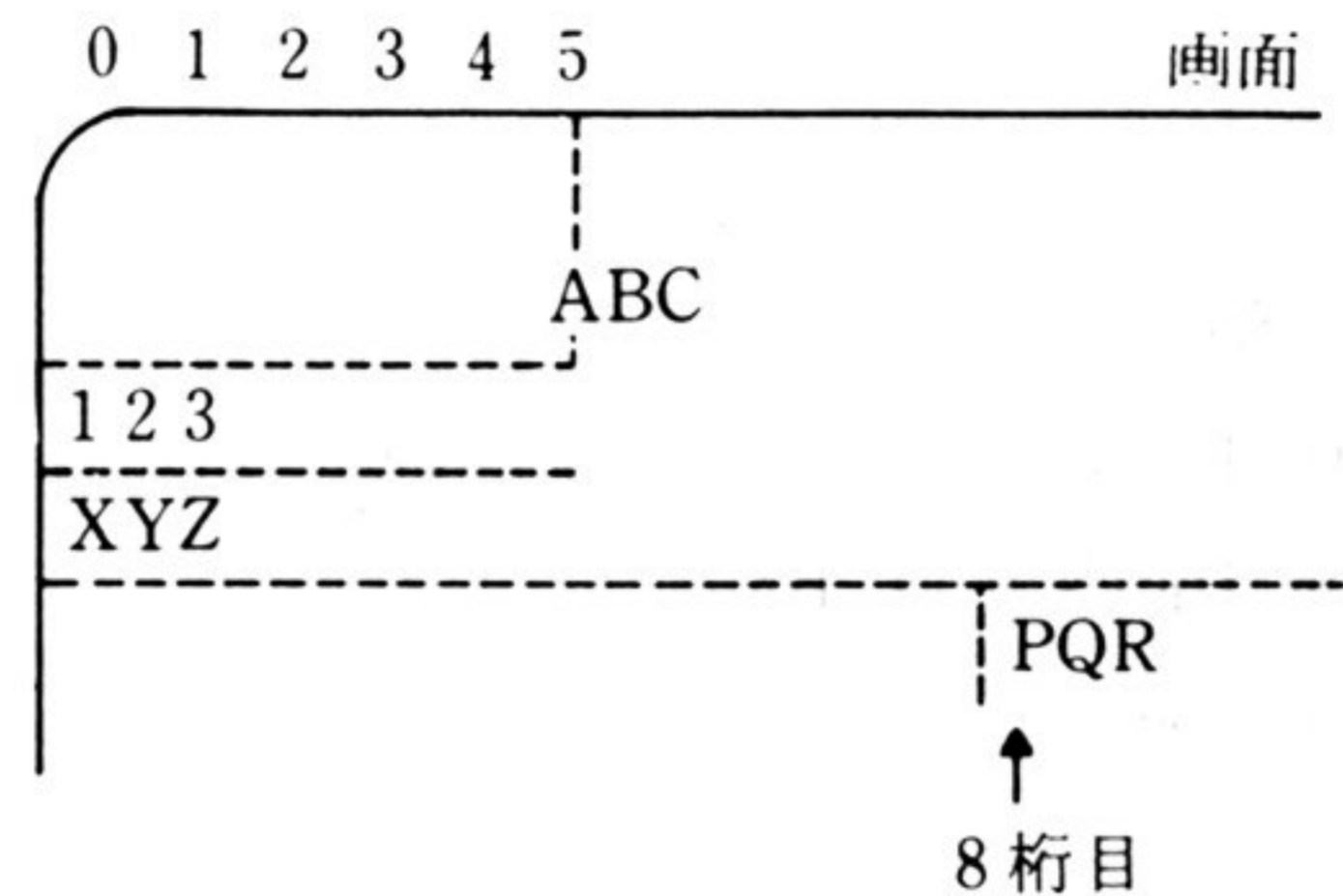
書き方

TAB（桁位置）

TAB関数を書く時の注意は，次のとおりです．

・桁位置には，式が指定できます．
・桁位置の値は，－32768～32767まで許されます．ただし，負の値は，0とみなされます．
・TAB関数が使用できるのは，PRINT文，LPRINT文の中だけです．

例

①PRINT TAB（5）；"ABC"
②PRINT TAB（80）；"123"
③PRINT "XYZ"，TAB（8）；"PQR"

上記例で①は指定された5桁目から「ABC」が，②は，画面の右端が79桁目なので，指定数字を80で割った余りが，TABの設定位置になります．

③は，「XYZ」を表示した後，コンマで次式が区切られているため，「XYZ」に14桁分が割り当てられますので，次のTAB（8）は，この行では無視され，次の行に対して働きます．

⑤FOR～NEXT文

FOR～NEXTで囲まれた命令の集まりを指定回数分，繰り返し実行します．

書き方

FOR 制御変数＝初期値 TO 終値〔STEP 増分〕
　　∫
　命令の集まり
　　∫
NEXT 制御変数

FOR～NEXT文を書く時の注意は，次のとおりです．

・初期値，終値，増分は数値式が指定できます．
・「ＳＴＥＰ増分」を省略すると，増分として１が指定されたものとみなします．
・ＦＯＲ～ＮＥＸＴ文の実行の仕方は，制御変数に初期値をセットし，ＮＥＸＴ文に達するたびに，増分だけふやします．その制御変数の終値を超えた時にＦＯＲ～ＮＥＸＴの繰り返しが終わり，ＮＥＸＴの次の文へ実行の流れが移ります．
・ＦＯＲ～ＮＥＸＴの繰り返しの中に，別のＦＯＲ～ＮＥＸＴの繰り返しを含むことができます．その際，内側のＦＯＲ～ＮＥＸＴと外側のＦＯＲ～ＮＥＸＴの制御変数は異なっていなければなりません．

例

①
```
10 FOR   I=1 TO 10 STEP 2
20 PRINT I;I^2
30 NEXT  I
```

この例では，Ｉを１から始めて10を超えるまで２つずつ増しながら繰り返しを行うので，FOR～NEXTの繰り返し回数は５回になります．

繰り返し行っている処理は，制御変数Ｉの内容とその２乗を画面に表示しているわけです．

②
```
10 FOR   I=1  TO  0
20 PRINT I
30 NEXT  I
```

この例では，初期値が終値を最初から超えていますから，ＦＯＲ～ＮＥＸＴの繰り返しを１回も実行せずに，ＮＥＸＴの次の文に実行の流れが移ります．

③
```
10 FOR   I=1  TO 9
20 FOR   J=1  TO 9
30 PRINT I ; "X" ; J ; "=" ; I * J
40 NEXT  J
50 PRINT
60 NEXT  I
```

画面
```
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
    ≀
1 × 9 = 9

2 × 1 = 2
2 × 2 = 4
    ≀
```

この例では，外側のＦＯＲ～ＮＥＸＴの繰

り返しを１回実行する間に，内側のＦＯＲ～ＮＥＸＴの繰り返しを９回実行します．これによって九九（クク）の計算式および結果が画面表示されています．

⑥ＩＮＴ関数

書き方

ＩＮＴ（数式）

指定された数式の整数部分だけを抽出します．

例

```
ＰＲＩＮＴ　ＩＮＴ（1003.56）
1003
```

第3章　シーケンシャルファイルの利用

本章では，フロッピィディスクにデータを記録していく手順や，記録したデータを入力し，処理するプログラムの作成方法を学びます．

本章のポイント

- シーケンシャルファイルの作成
- シーケンシャルファイルのデータ入力
- 画面の利用の仕方

第1節　データのフロッピィディスクへの登録

第1節では，身近な例として家計簿をとりあげてみましょう．

家計簿は，1カ月前，半年前，更には昨年と比較してみるとどこに浪費があるのか，何が昨年より使いすぎかといったことが分り，節約のポイントがつかめます．

ここではその第1歩である，毎日の収支データの家計簿への記入です．コンピュータでこの仕事を行う場合，フロッピィディスクという家計簿へ，キーボードというペンで，データを記入（蓄積）するプログラムを作成することになります．

習得ポイントは，ディスプレイ画面を見ながらデータを入力する方法と，フロッピィディスクへのデータの登録です．

1　家計における収支データの作成（例題3－1）

例題の概要

これから考えていただく例はディスプレイ表示画面を見ながら，毎日の収入，支出データをキーボードから入力し，収支データとしてフロッピィディスクに登録するプログラムです．

画面表示形式

◎フロッピィに登録する収支データの1件分のデータの形式は，次のとおりです．

各々の項目は，キーボードから入力されるデータをフロッピィディスクへそのまま出力することにより作成できますが，金額については，数量と単価の積を出力します.

処理の条件

○データの入力は表示画面に従って行います.

○キーボードから入力する各々のデータ項目は，次のような条件で入力します.

・収支コード……………………………0　収入 ｝いずれか1桁の入力
　　　　　　　　　　　　　　　……1　支出 ｝
　　　　　　　　　　　　　　　……9　処理の終了時

・日付……………………………………月日を各々2桁ずつ入力

・区分コード……右の表より該当する項目のコードを選択し，入力します.

区分	コード
収入	0
食費	1
光熱費(電気，ガス，水道)	2
衣料費	3
教養，娯楽費	4
教育費	5
交際費	6
医療費	7
衛生費	8
住居費	9
貯金（預金）	10
保険費	11
電話代	12
雑費	13

・項目名……………各収支の詳細内容(20桁以内)

・単位………………各項目の単位

・数量………………各項目の数量（数えられないものも1とします）

・単価………………各項目の単価（数えられないものは金額と同じ値にします）

○データ入力件数は1画面で9件まで（画面上21行目）とします．9件目を入力したあと，引き続き画面を表示しておくため，23行目で次のような入力処理を行います．すなわち，23行目に「カクニン」と画面表示が出た時点で，次にとるべき行動の，該当文字を入力します．そしてプログラムでは，その入力文字に対する処理を行います（この処理動作を確認入力といいます).

次にとるべき行動	入力文字	プログラム上での処理
現在表示されているデータをまだ見たい		（そのまま）
残りのデータを入力したい	C	現在表示されている画面を消し，続くデータを5行目から入力できるようにします

入力すべきデータがないのでプログラムを終了したい	E	プログラムを終了させます
	CまたはE以外	確認入力の再実行

○プログラムの終了は，収支コード項目で「9」が入力されるか，確認入力で「E」が入力された時とします．

ことば

確認入力

確認入力とは，ディスプレイ画面を使っているプログラムで，現在，画面に表示されている内容を別の画面に切り替えていいかどうかを判断するための入力動作です．更に切り替える画面が何種類もある場合，どの画面（実際には，どんな処理動作を行うか）かを判断するために先の例のようにCとかEとかの文字を入力している訳です．又，それとともに，何か文字が入力されるまでは，現在の画面が表示されていますので，表示画面を見ながら入力データのエラーチェックもできる訳です．

プログラム作成上の注意点

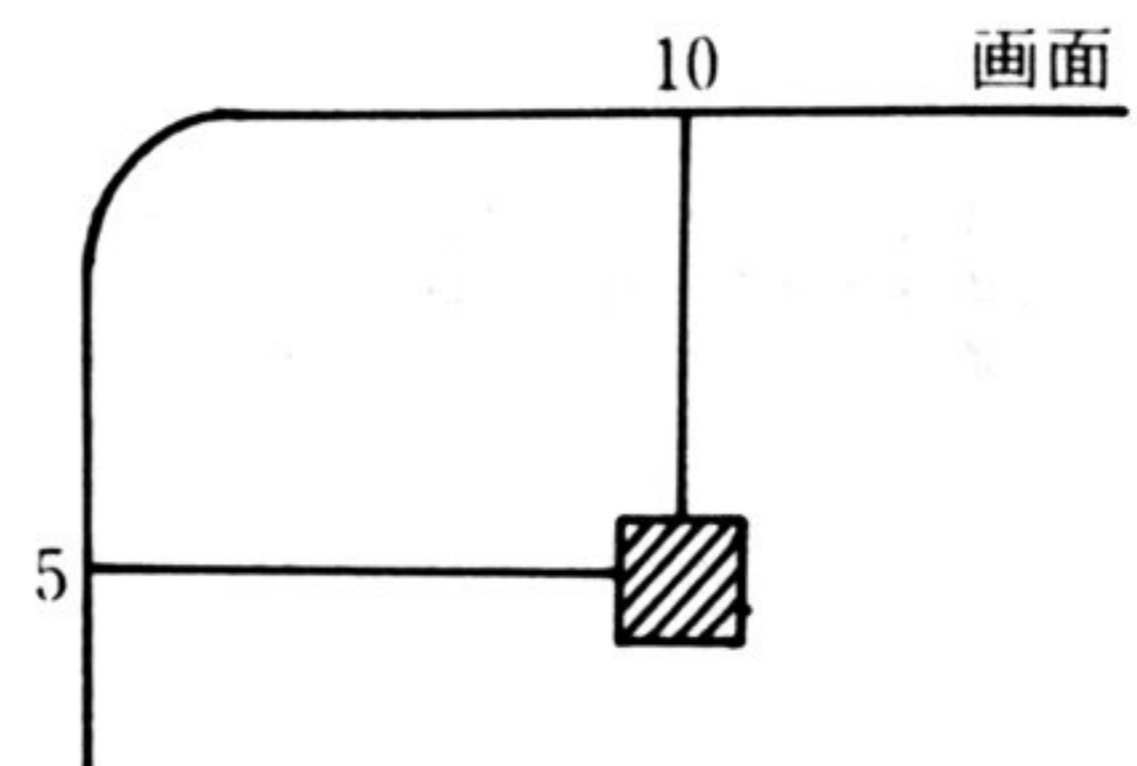
このプログラム作成上のポイントとして，画面上の決まった位置からデータの入力や表示を行っているという点があげられます．

入力するデータ量が多い場合には，1章で学んだような方法でなく，画面上の配置（画面レイアウト）をしっかり決めた方法でデータ入力を行った方が操作性が向上します．BASICにおいては，画面上の特定の場所に位置づける命令として，「LOCATE（ロケート）」文やTAB関数が用意されています．

TAB関数については，2章で既に説明しました．LOCATE文は，次のように指定できます．

LOCATE　桁位置，行位置

たとえば，「LOCATE 10 ,5」と指定すると，右図のように画面上の希望する場所に位置づけが行えます．

ところで2章で説明しましたが，1件分の処理手順だけを考えてやり，あとは，その命令を繰り返して実行させるというのがプログラムの基本型でした．今回取り扱うプログラムも，画面上の各入力行が全く同じ形式なので繰り返しが可能であるというのはお分かりいただけるでしょう．しかし，各データを入力する桁位置が同じであっても，1件分すなわち1行入力するたびに行位置が変わっていきますから，上記のような形で命令を書くことはできません．そこで，今回の例のような場合は，行位置が変更できるように，次のような命令の書き方をする必要があります．

LOCATE　桁位置，行位置を表わす変数
　　　　　　　　　　↑変数なので値が変更できます．

[2] プログラム例

次にプログラム例を示します．実際にプログラムをキーボードにより入力し，実行して見て下さい．

```
10 OPEN "1:SYUSHIF" FOR OUTPUT AS #1
20 PRINT CHR$(&HC)
30 LOCATE 17,1:PRINT "*** ｼ ｭ ｳ ｼ ﾃﾞ - ﾀ ﾆ ｭ ｳ ﾘ ｮ ｸ ***"
40 LOCATE  2,3:PRINT "ｼｭｳｼ  ﾋﾂﾞｹ   ｸﾌﾞﾝ    ｺ ｳ ﾓ ｸ ﾒ ｲ"
50 LOCATE  2,4:PRINT "ｺｰﾄﾞ          ｺｰﾄﾞ"
60 LOCATE 45,3:PRINT "ﾀﾝｲ     ｽｳﾘｮｳ   ﾀ ﾝ ｶ    ｷ ﾝ ｶﾞ ｸ"
70 LIN=5
80 LOCATE  2,LIN:INPUT IDENT%
90 IF IDENT%=9 THEN 240
100 LOCATE  6,LIN:INPUT HIZUKE%
110 LOCATE 13,LIN:INPUT KUBUN%
120 LOCATE 20,LIN:INPUT ITEM$
130 LOCATE 43,LIN:INPUT TANI$
140 LOCATE 53,LIN:INPUT SURYO%
150 LOCATE 59,LIN:INPUT TANKA#
160 KINGAK#=SURYO%*TANKA#
170 LOCATE 70,LIN:PRINT KINGAK#
180 PRINT# 1,IDENT%;HIZUKE%;KUBUN%;ITEM$;",";TANI$;",";SURYO%;TANKA#;KINGAK#
190 IF LIN=21 THEN 200 ELSE 230
200 LOCATE 61,23:INPUT "ｶｸﾆﾝ  ";CHK$
210 IF CHK$="C" THEN 20
220 IF CHK$="E" THEN 240 ELSE 200
230 LIN=LIN+2:GOTO 80
240 CLOSE:END
```

それでは，プログラムを分けて見ていきましょう．

①

```
10 OPEN "1:SYUSHIF" FOR OUTPUT AS #1
20 PRINT CHR$(&HC)
30 LOCATE 17,1:PRINT "*** ｼ ｭ ｳ ｼ ﾃﾞ - ﾀ ﾆ ｭ ｳ ﾘ ｮ ｸ ***"
40 LOCATE  2,3:PRINT "ｼｭｳｼ  ﾋｽﾞｹ   ｸﾌﾞﾝ    ｺ ｳ ﾓ ｸ ﾒ ｲ"
50 LOCATE  2,4:PRINT "ｺｰﾄﾞ          ｺｰﾄﾞ"
60 LOCATE 45,3:PRINT "ﾀﾝｲ     ｽｳﾘｮｳ   ﾀ ﾝ ｶ    ｷ ﾝ ｶﾞ ｸ"
```

10～60行は，準備作業を指示するためのものです．

10行目では，フロッピィディスクに収支データを書き出す前の準備を行っています．各指定は次のとおりです．

20行目は，画面全体の消去（画面クリア）の指定です．

30～60行は，画面への見出しの表示を行っています．たとえば30行では，LOCATE文によって１行17桁に位置づけPRINT文で引用符の中の定数を表示しています．他の行も同様です．

```
70 LIN=5
80 LOCATE  2,LIN:INPUT IDENT%
90 IF IDENT%=9 THEN 240
100 LOCATE  6,LIN:INPUT HIZUKE%
110 LOCATE 13,LIN:INPUT KUBUN%
120 LOCATE 20,LIN:INPUT ITEM$
130 LOCATE 43,LIN:INPUT TANI$
140 LOCATE 53,LIN:INPUT SURYO%
150 LOCATE 59,LIN:INPUT TANKA#
160 KINGAK#=SURYO%*TANKA#
170 LOCATE 70,LIN:PRINT KINGAK#
```

70行でLINという変数に５を与えていますが，何をしているのでしょう．80行，100～150行，170行でLOCATE文の行位置指定の所でLINという変数を使っていますね．この値を最初に５から始めたいための処理なのです．80行で，LINの値が５ですから，５行２桁目から「IDENT%」に収支コードを入れさせているのです．画面表示形式で確認して下さい．実際には「？」の位置になっているはずです．同様に100行～150行も同じ動作を行います．

160行ではキーボードから入力した数量（SURYO%）と単価（TANKA#）をかけ金額（KINGAK#）を計算し，170行で表示しているのです．このLINの内容を２ずつふやすことによって，行を変えながら，同じ場所（桁）から入力が続けられることになります．

さて90行ですが，ここでは，IDENT%に「９」という数字が入力されたかどうか調べています．つまり「IF（イフ）」という命令は条件（判断）命令です．この例の処理条件にもあったように，収支コードで「９」が入力されたならプログラムを終わらせるという決まりがありました．そこで，変数IDENT%に９が入っていれば，240行へ実行を移し，そうでなければ100行に移るよう命令が指定されています．

```
180 PRINT# 1,IDENT%;HIZUKE%;KUBUN%;ITEM$;",";TANI$;",";SURYO%;TANKA#;KINGAK#
190 IF LIN=21 THEN 200 ELSE 230
200 LOCATE 61,23:INPUT "カクニン  ";CHK$
210 IF CHK$="C" THEN 20
220 IF CHK$="E" THEN 240 ELSE 200
230 LIN=LIN+2:GOTO 80
240 CLOSE:END
```

160行までの処理によって1件分のデータができあがりますから，180行で「PRINT #(プリント)」文を使って8項目を1まとめ（1件のデータ）としてフロッピィディスクに書き出しています．

190行では，変数LINの内容が21かどうか調べて，条件が満足していれば200行，そうでなければ230行に実行を移すようにしています．これは，画面上での入力行が21行目に達しているかどうかを調べているわけです．21行に達していれば，確認入力を行う必要があるので，200行で画面上23行61桁に位置づけ，確認入力を行わせるようにしています．

この確認入力で，何が入力されたかを調べているのが210行，220行です．210行で確認キーとして「C」が押されたかどうかを調べ，押されていれば20行へ実行を移し，それ以外のキーが押されていたならば220行に実行を移します．220行では，確認キーとして「E」が押されたかどうかを調べ，押されていれば240行へ，それ以外のキーが押されていたならば200行へ戻り，確認キーの入れなおしを行わせています．

LINの内容が21に達していない時には，同じ画面で入力が続けられますから，230行で現在のLINの内容に2を加算して，80行に実行を戻します．この作業によって，前のデータ入力行から2行下の位置から入力が続けられるわけです．

240行は，確認入力で「E」が入力された時の処理，プログラムの終了処理です．「CLOSE(クローズ)」文で，使用したフロッピィディスクデータの後始末を行っています．次の文「END」でプログラムの終了になります．

注意

IF文の書き方は次のようになっています．

ELSE以降が省略されている時は，次の文が実行されます．

注意

180行目のPRINT#で，フロッピィディスクにデータを出力していますが，その際，8つの項目の他に，ITEM$とTANI$の後に続けて「,」を書き足しています．どうして「,」を書き足したのでしょうか．実は，コンマを指定しなかった場合は，次のようにデータが記録されます．

△1△△920△△1△カキノタネフクロ△1△△400△△400△

数値データの場合は，必ず前後に空白がとられますが，文字データにはそのようなことは行われていないのです．つまり，データを登録する際には何ら支障はなくても，登録されたフロッピィ上のデータを入力してくる時に，項目の区切りがないために，項目名以降のデータが全て項目名として入力されてきてしまい，まったく別の意味のデータとなってしまいます．

△1△△920△△1△カキノタネフクロ△1△△400△△400△

そのため，このような不都合をなくすためPRINT#でフロッピィディスクにデータを出力する際は，文字データの後に必ず「,」を書き出すようにします．

③ 処理の手順

プログラム例をフローチャートで考えてみましょう．

このプログラムも，プログラムの基本型に従って実行手順が考えられています．

プログラムの最初と一番最後で実行している「ファイル準備」，「ファイル後始末」は，フロッピィディスクのデータを取り扱う時には，必ず実行しなければならない処理です．これを実行しないと，フロッピィディスクにデータを書き出したり，逆にデータを入力することはできません．

また，画面全体を使ってデータ入力，表示を行う場合は，最初に画面全体の消去を行う必要があります．この処理をしておかないと，以前に画面に表示されていた内容と表示内容が重なってしまって意味不明の画面となってしまいます．

準備が終わったならば，画面左側より順番にデータを入力するようにします．

今回，計算処理としては金額計算があります．

出力としては，計算した金額を画面に表示する作業は，入力されたデータと金額をあわせて1件のデータとしてフロッピィに書き出す処理があります．

今回のプログラムでは，この一連の入力→処理→出力をデータ入力件数分だけ繰り返していくことになります．

ただし，2章の例と違い，一画面上での繰り返しの制限（9件分の表示）がありますから，それを調べる動作が当然のことながら必要になってきます．

それと関連する処理が右図の「画面切換処理」の部分です．

画面右下23行目で「確認入力」という方法を用いて，新しい画面への切換え，もしくはプログラムの終了検査を行っています．

プログラムの処理手順は，基本的には，入力→処理→出力という処理の繰り返しですが，コンピュータに行わせる仕事が複雑になってくると，それに付随する多くの処理が必要になってきます．

しかし，プログラム自身はできるだけ簡単に作ることが，いろいろな意味からも重要なことです．

また，この例では，プログラムを終了させる判断を2ヶ所で行っています．確認キーとして「E」が入力された時，収支コードとして「9」が入力された時です．

プログラムを終了する場面は，会話型のプログラム（画面を使って入出力を行っているプログラム）においては，いろいろ考えられます．そのため，プログラムを作る場合は常に，い

ろいろな場面を想定して，処理のもれがないようにしておかなければなりません．

4　命令の説明

この節で新しく出てきた命令を，見ていくことにしましょう．

① OPEN文

ファイルを使用できる状態にします．

書き方

OPEN“ファイルディスクリプタ”〔FORモード〕AS〔#〕ファイル番号

OPEN文を書く時の注意は次のとおりです．

○ファイルディスクリプタでは，これから作成するファイル名，既に作成済みのファイルからデータを入力しようとするファイル名等を指定します．

○FORで指定するモードの種類と意味は，次のとおりです．

INPUT………既に作成済みのファイルからデータを入力する時の指定
OUTPUT……新しくファイルを作成する時の指定
APPEND…… 既に作成済みのファイルにデータを追加登録する時の指定

○ファイル番号としては1～15が指定できます．OPEN文以降の入出力命令では，ファイル番号で各ファイルを区別して取り扱っていきます．

② CLOSE文

使用したファイルの後始末を行います．

書き方

CLOSE〔〔#〕ファイル番号〔,〔#〕ファイル番号〕・・・〕

CLOSE文を書く時の注意は次のとおりです．

○ファイル番号で指定されたファイルの後始末を行います．

○CLOSEのみの指定の時は，その時にOPENされているすべてのファイルをCLOSEします．

③ LOCATE文

画面上の特定の位置にカーソルを位置づけします．

書き方

LOCATE〔桁位置〕〔, 行位置〕

LOCATE文を書く時の注意は，次のとおりです.

○桁位置が省略された時は0，行位置が省略された時は，現在カーソルがある行が指定されたものとみなされます.

④ PRINT#文

シーケンシャルファイルにデータを出力します.

書き方

PRINT# ファイル番号,〔式〔{;,} 式〕・・・〕

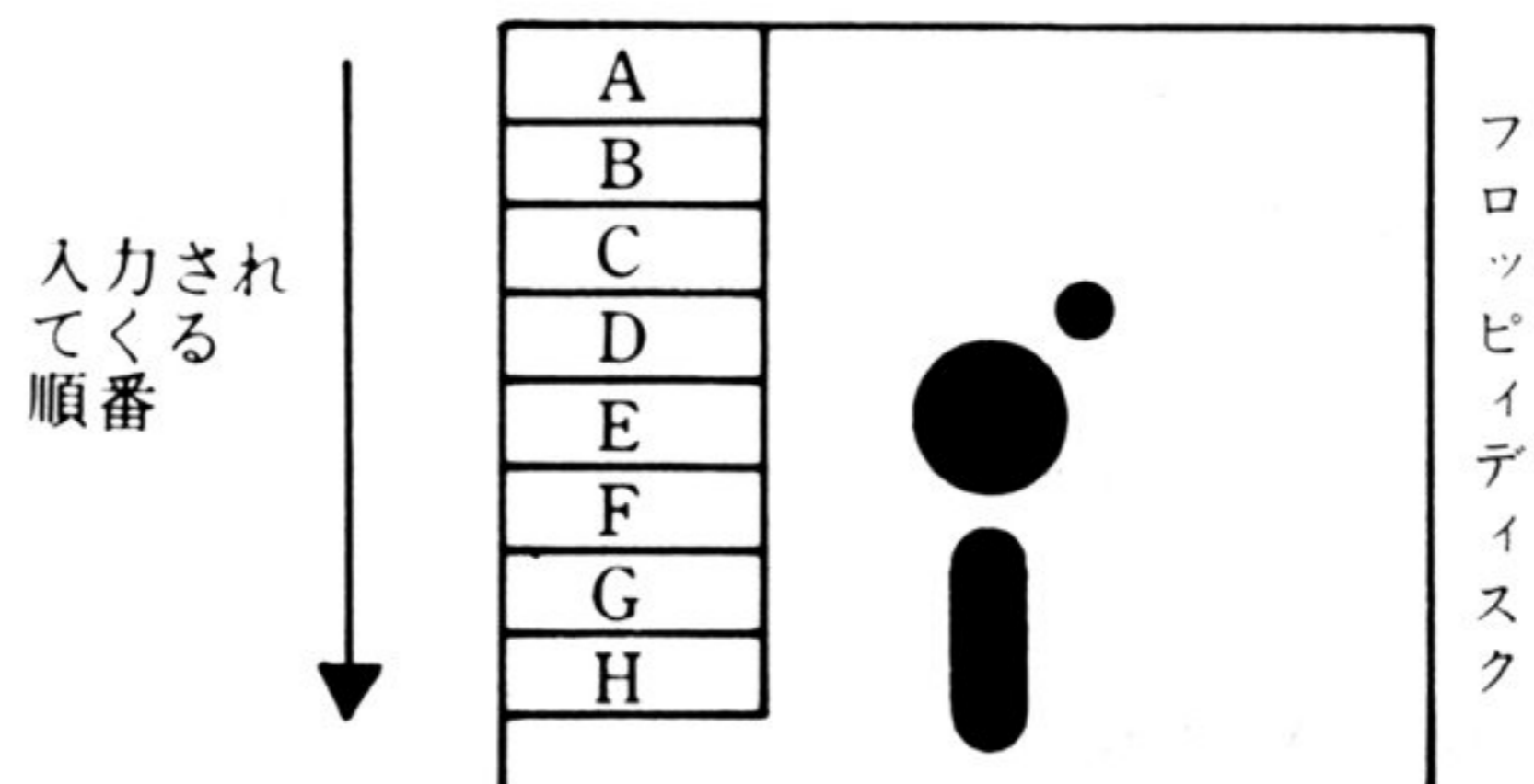
PRINT#文を書く時の注意は，次のとおりです.

○式と式との間に「;」を使うか,「,」を使うかによってPRINT文の時と同様の領域の割り当てが行われます.

○文字データを書き出す際は，その直後に必ず「,」を書き出して下さい.

ことば

シーケンシャルファイル

BASICで扱えるファイルには，シーケンシャルファイルとランダムファイルがあります．ここで取り扱っているシーケンシャルファイルは，データを登録した順番に記録していき，入力してくる際も，記録されている順番に入力できるファイルです.

⑤ IF文

条件を満足しているかどうか調べ，それに応じて命令を実行させます．

書き方

IF 条件 THEN { 行番号 ／ 文 } 〔ELSE { 行番号 ／ 文 }〕

IF文を書く時の注意は，次のとおりです．

○条件を満足した時は，THENの後の文を実行するか，または指定された行番号に実行を移します．

○条件を満足しない時は，ELSE以降の文を実行するか，または指定された行番号に実行を移します．ELSE以降が省略されていた時は，次の文を実行します．

○条件の中では，次のような関係演算子や論理演算子が使えます．

関係演算子	
=	
<>, ><	……等しくない
<	
>	
<=, =<	……以下
>=, =>	……以上

論理演算子	
NOT	……否定
AND	……論理積
OR	……論理和
XOR	……排他的論理和
IMP	……包含
EQV	……同値

例

(1) IF A＝0 AND B<＝0 THEN C＝C＋1 ELSE C＝C＋2

上の文はAが0でかつBが0以下の時，THEN以降の文（C＝C＋1）を実行します．Aが0ではないか，Bが0をこえる時は，ELSE以降の文（C＝C＋2）を実行します．

(2) IF A＝1 OR B<>1 THEN C＝C∧2

上の文はAが1またはBが1でない時，どちらか一方が満足した時にC＝C∧2を実行します．

⑥ GOTO文

実行の流れを変更します．

書き方

GOTO　行番号

○GOとTOの間に，空白を1個入れてもかまいません．

⑦　CHR$関数

指定したコードを持つ文字を与えます．

書き方

CHR$（数式）

○数式の値と対応する文字については，ユーザーズマニュアルの付録を参照して下さい．

第2節　フロッピィディスク上のデータ入力

第1節でフロッピィディスクに登録した収支データを入力して，プリンタに印字するプログラムを通し，ファイルからのデータ入力について学習します．

① 収支データ表の作成（例題3－2）

例題の概要

第1節のプログラムによってフロッピィディスクに登録されている収支データを入力し，次の形式でプリンタに印字させます．

処理の条件

○フロッピィディスクに登録されている収支データのすべてをプリンタに印字します．

○プログラムの終了は，全データを印字し終った時とします．

プログラム作成上の注意

このプログラムは，それほど複雑ではありません．収支データを1件ずつフロッピィより入力し，それを1行ずつ印字していけばよいわけです．ただし登録のさいに「収支コード」もファイルに出力しましたが，この内容は印字しません．

本プログラムでの注意点は，プログラムをいつ終わらせるかということです．「例題の概要」で，全データを印字し終った時にプログラムを終了させるとあります．では，全データを印字し終ったかどうかはどのように判断したらよいのでしょう．

最後のデータを印字しても，それが一番最後のデータかどうかは，データを記録した時に覚えておかない限り，まずわからないでしょう．

BASICでは，データを記録した時に最後のデータの次に終わりマーク(EOF…End of File)をつけることによって，このマークが入力された時にデータが終わりであるということが判別できるようになっています．

② プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．

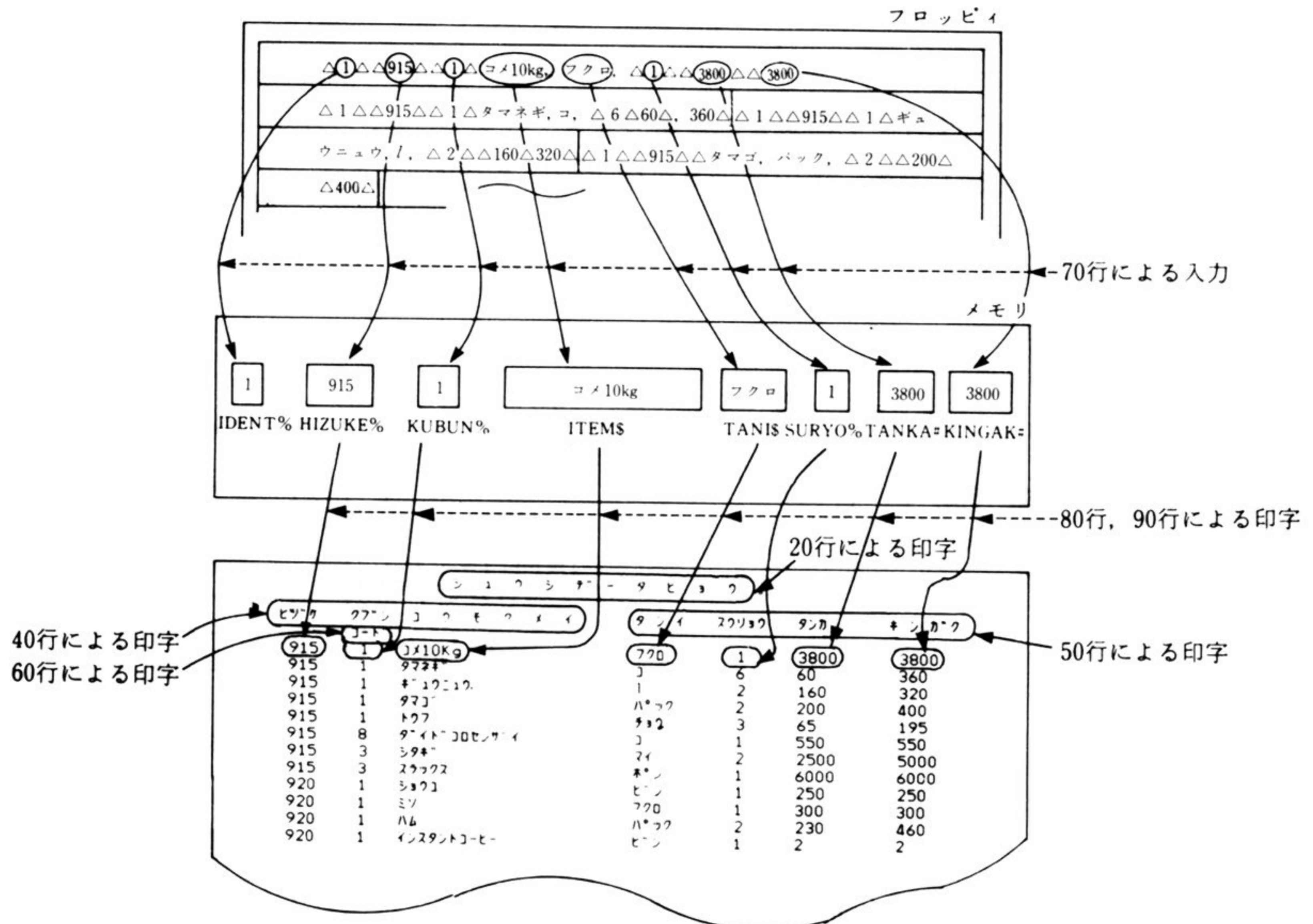
```
10 OPEN "SYUSHIF" FOR INPUT AS 1
20 LPRINT TAB(21);"ｼ ｮ ｳ ｼ ﾃﾞ - ﾀ ﾋ ｮ ｳ"
30 LPRINT
40 LPRINT TAB(4);"ﾋｽﾞｹ   ｸﾌﾞﾝ ｺ ｳ ﾓ ｸ ﾒ ｲ";
50 LPRINT TAB(39);"ﾀ ﾝ ｲ   ｽｳﾘｮｳ   ﾀﾝｶ     ｷ ﾝ ｶﾞ ｸ"
60 LPRINT TAB(11);"ｺｰﾄﾞ"
70 INPUT# 1,IDENT%,HIZUKE%,KUBUN%,ITEM$,TANI$,SURYO%,TANKA#,KINGAK#
80 LPRINT TAB(4);HIZUKE%;TAB(11);KUBUN%;TAB(16);ITEM$;TAB(39);TANI$;
90 LPRINT TAB(48);SURYO%;TAB(54);TANKA#;TAB(64);KINGAK#
100 IF EOF(1) THEN 110 ELSE 70
110 CLOSE:END
```

まず10行目でファイルの準備を行っています．FORの後に「INPUT」が指定されていますから，既に作成済みのファイルからデータを入力してくることを意味しています．「収支ファイル（SYUSHIF）」に対して，ファイル番号「１」が対応づけられていますね．

20行目で大見出しの印字，30行目で空白行の印字，40，50，60行目で項目見出しの印字を行っています．

以上の準備作業をすべて行った後，70行目で，データを１件分入力してきます．ファイル番号１の後に指定されている「IDENT%」から，「KINGAK#」までの変数にデータを入力してくるわけです．

そして，入力したデータを80行，90行の「LPRINT」文を使って所定の位置へ印字しています．あとは，これを全データに対して繰り返せばよいわけです．繰り返し行うか，それとも入力してくるデータがなくなったので，プログラムを終わりにするかどうかの判断が次の100行で行われています．

100行では，１番のファイル（SYUSHIF）がEOF（ファイルの終わり）まで達しているかどうかを調べ，達していれば最後のデータまで印字が終了したということで110行に移りプログラムの終了になります．まだ印字の済んでいないデータがあれば，70行へ戻し次のデータを入力してきます．

③ 処理の手順

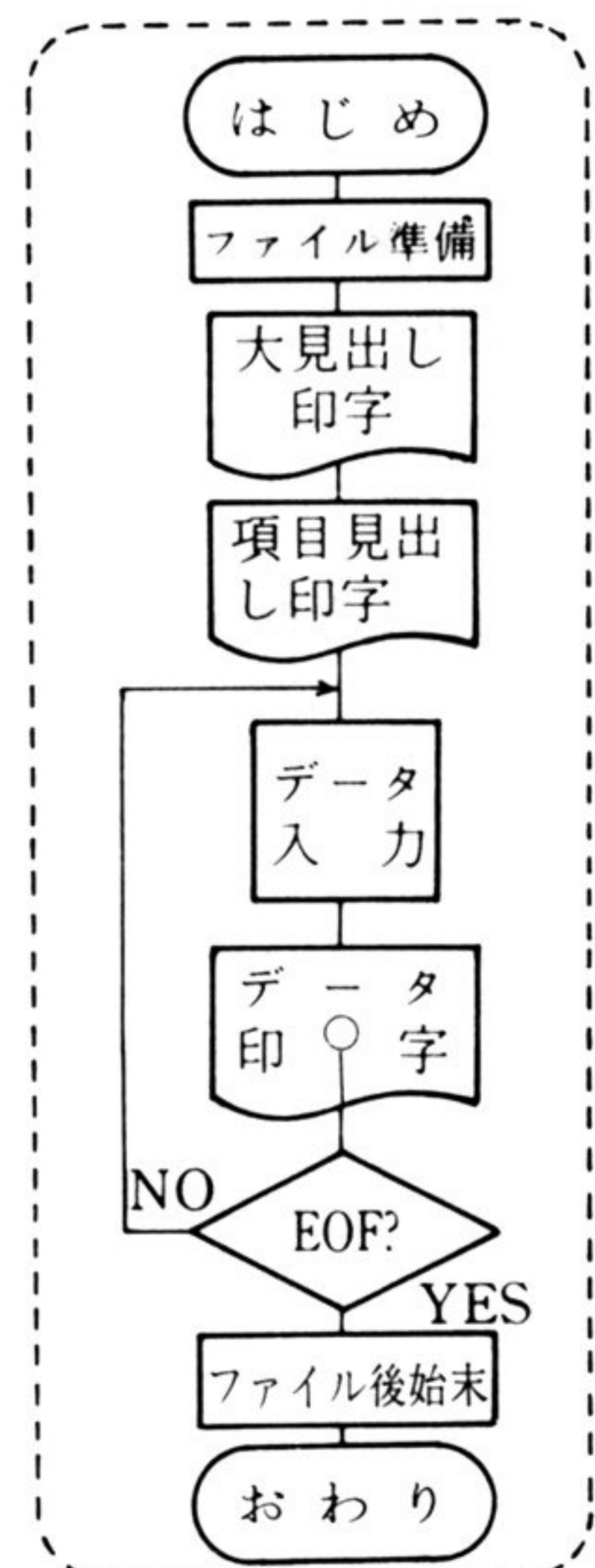
プログラム例をフローチャートで考えてみましょう．右図の手順からもわかるように，準備―入力―（処理）―出力―後始末の基本的な処理の形であることは確認できますね．基本的なパターンと異なる点は，プログラムを終了にするかどうかの判断の位置が異なることです．通常は，データ入力の直後で入力データがなくなったかどうかの判定を行いますが，今回は，データ出力（印字）の直後に終了判定を入れています．

これは，BASICのファイル入力命令（INPUT#）が，データ入力を行った後，次のデータの先頭に位置づけることが原因です．もし，終了判定がデータ入力の直後にあると，最後のデータを入力し，それを印字しないうちにプログラムが終わりになるからです．

④ 命令の説明

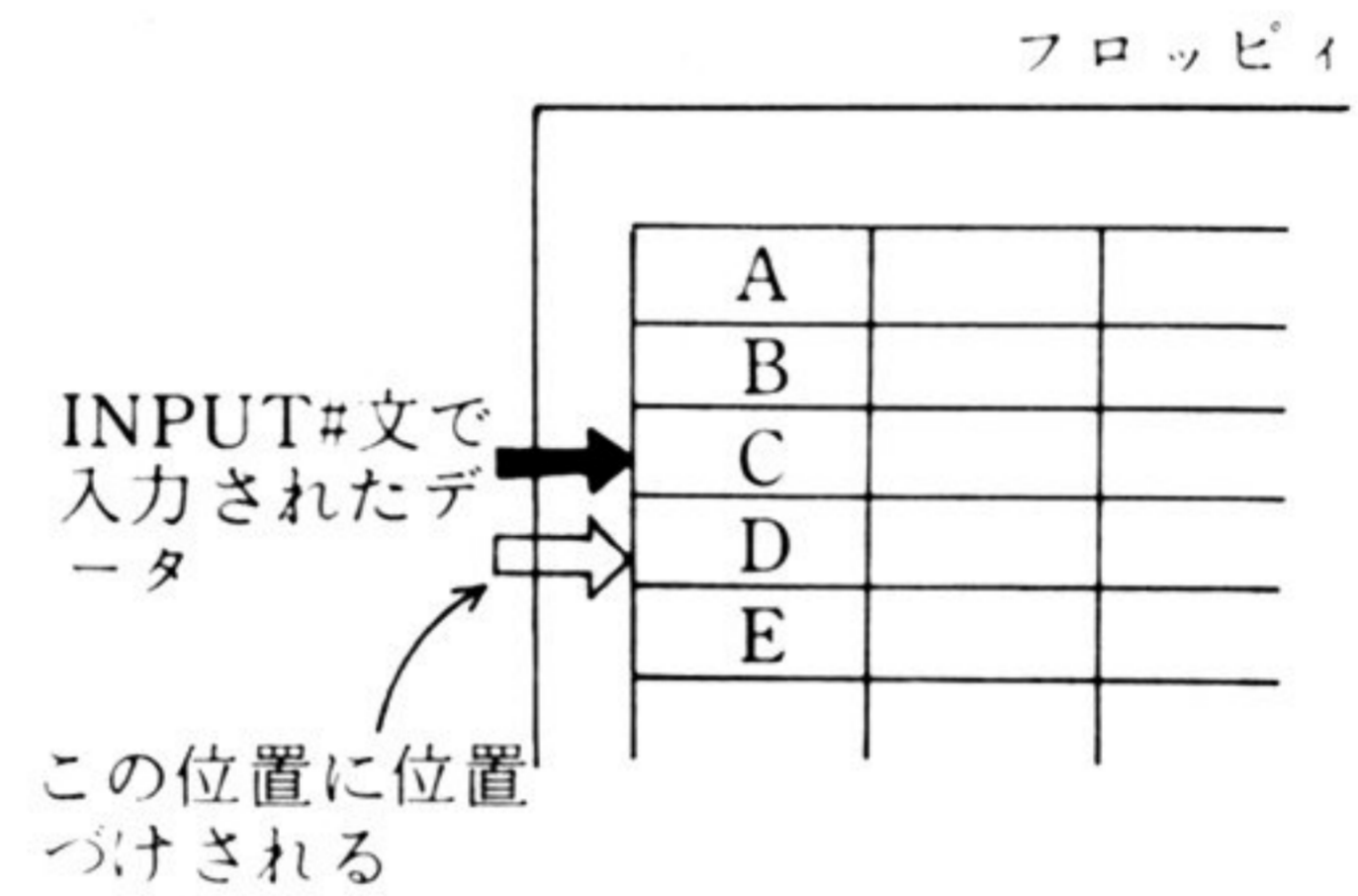
この節で新しく出てきた命令を見ていくことにしましょう．

① INPUT#文

シーケンシャルファイルからデータを入力していきます．

書き方

INPUT#　ファイル番号，変数名〔，変数名〕・・・

INPUT#文を書く時の注意は次のとおりです．

○ファイル番号は，OPEN文で指定された番号です．
○INPUT#でデータを入力後，次のデータに位置づけされます．

② EOF関数

ファイルの終りを検出します．

EOF（ファイル番号）

ファイル番号で指定されたファイルの中のデータが，終わりになったかどうかを検出します．

第4章　見易い表の作成

本章では，見易い帳表の作り方を中心にケイ線でかこんだ表形式のプリント出力，サブルーチンを使用して，プログラムを簡潔にする方法について学習します。

本章のポイント

- 多くの変数の確保の仕方
- FOR～NEXTの多重ループの利用
- ケイ線を含んだ表の作成
- サブルーチンを使用してプログラムを簡潔に表現する方法

第１節　表形式のプリント

第１節では，公共料金使用推移表の作成プログラムを通して，ケイ線で枠をかこんだ表形式のプリントの印字方法について学習します．

① 公共料金使用推移表の作成（例題４－１）

例題の概要

次のような公共料金使用実績を基に，公共料金使用推移表を作成するプログラムを考えます．

公共料金使用料

項＼月	1	2	3	4	5	6	7	8	9	10	11	12
電気(KWh)	350	370	330	360	345	352	368	375	351	340	320	380
ガス(m′)	68	70	63	58	57	55	53	59	60	58	62	67
水道(m′)	63	57	58	60	61	53	57	65	60	58	57	60

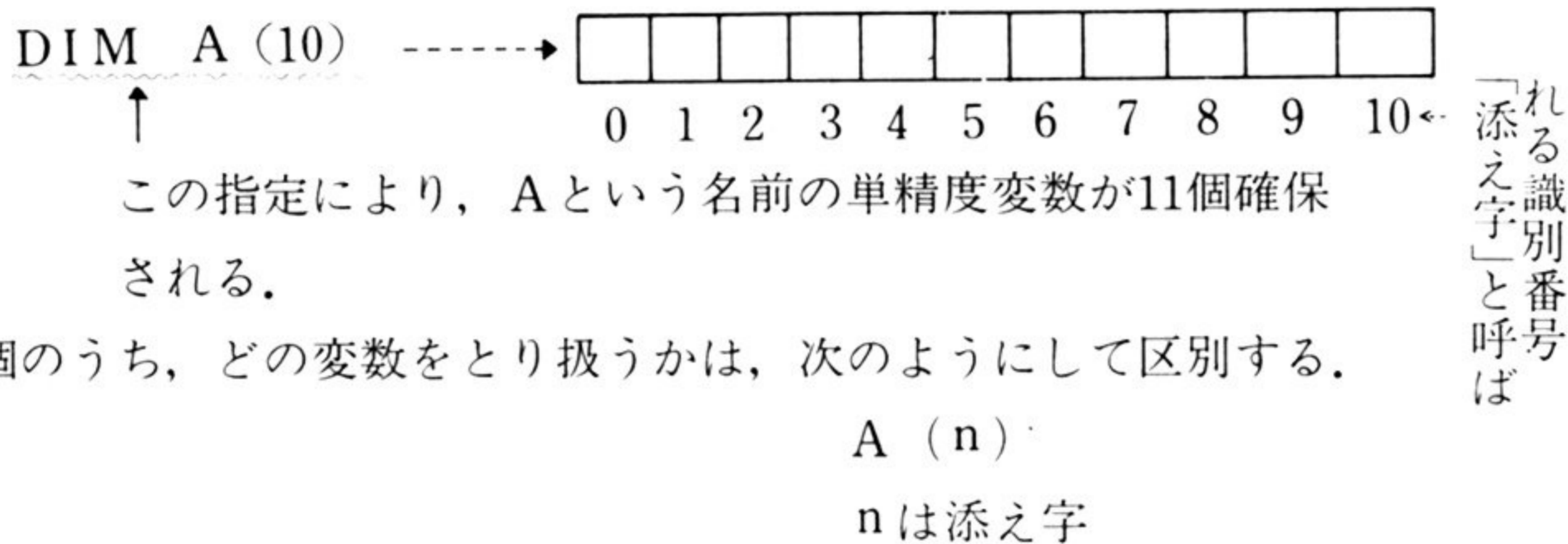
◎　公共料金使用推移表の印字形式

「###」で示される位置に実績値を印字します．各項の実績値の下には，括弧でくくって合計値に対する比率を％で印字します．この比率は，四捨五入して整数部分のみを印字します．

処理の条件

・実績値，比率ともに印字場所に数字の右側をそろえて印字します．

プログラム作成上の注意点

このプログラムは，各料金の合計値に対する各月の使用値を比率で表現する訳です．この処理を行うための方法はいろいろありますが，今回はDATA文で定義した実績値を，対応する36個（12ケ月×３種類）の変数に入力し，それを比率計算に使っています．ところで，36個もの変数を確保するとなると名前をつけるだけでも大変ですね．そこでBASICにおいては，同一属性の変数を数多く確保することが簡単にできる命令が，用意されています．この機能をうまく使うことによって，FOR～NEXT文を利用して大量のデータを簡単に取り扱うことができます．

また，このプログラムでは同じ種類のケイ線を何回も引いたり，同じパターンの印字を繰り返したりなどの，同一の処理がプログラム中で何回もでてきます．このような場合，その処理の必要な都度，命令を書くのではなく，プログラム内に，１ケ所だけ定義しておき，必要に応じてその処理を呼びだして実行させるという考え方をとります．この考え方を使用することにより，プログラムが簡潔になり，プログラムをより短くすることができます．

このように，プログラム中で何度も出てくる処理を１ヶ所にまとめたものを「サブルーチン」と呼び，サブルーチンを実行させる命令（GOSUB）が書かれている処理の流れを「メインルーチン」と呼びます．

メインルーチン
GOSUB 500
GOSUB 500
GOSUB 500
500
（500行から先がサブルーチン）
▨の部分で全く同じ処理が行われていた場合

② プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．

メインルーチン（10～370行）: DATAの配列変数への入力（50～90行）／見出し印字（100～140行）／合計印字（300～360行）
サブルーチン1（380～420行）
サブルーチン2（430～470行）
サブルーチン3（480～530行）

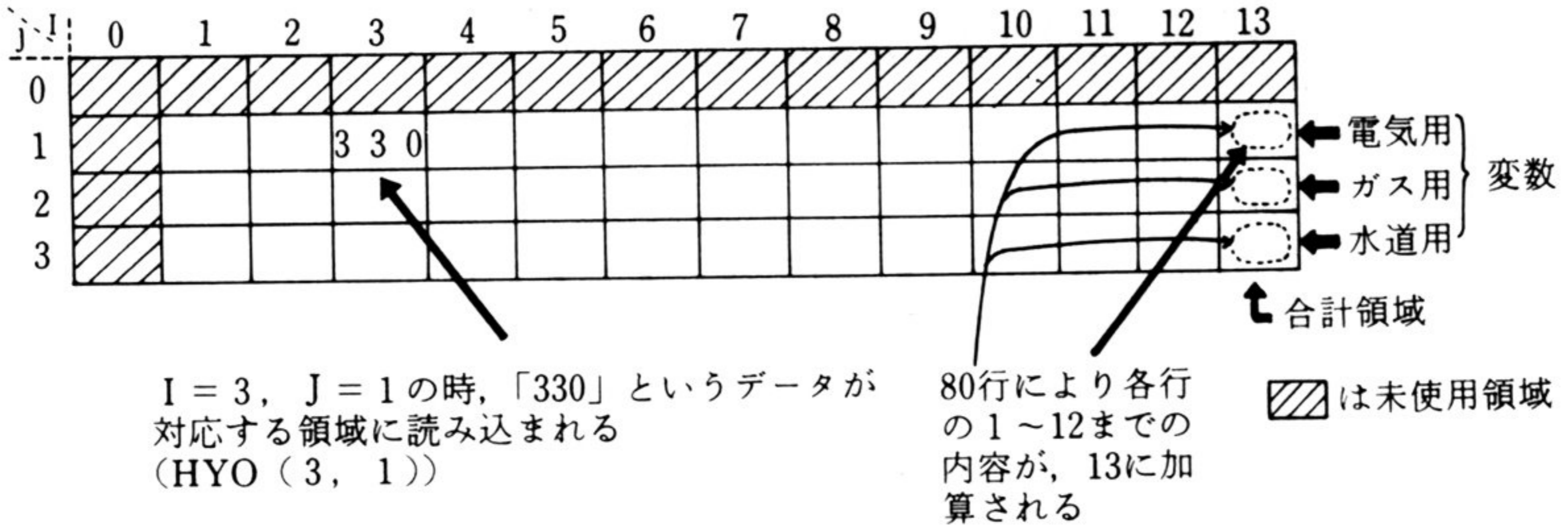
```
10 DATA 350,370,330,360,345,352,368,375,351,340,320,380
20 DATA 68,70,63,58,57,55,53,59,60,58,62,67
30 DATA 63,57,58,60,61,53,57,65,60,58,57,60
40 DIM HYO(13,3)
50 FOR J=1 TO 3
60  FOR I=1 TO 12
70  READ HYO(I,J)
80  HYO(13,J)=HYO(13,J)+HYO(I,J)
90 NEXT I,J
100 LPRINT TAB(29);"コ ウ キ ョ ウ リ ョ ウ キ ン シ ヨ ウ リ ョ ウ"
110 LPRINT
120 LPRINT "   ツキ     1     2     3     4     5     6     7     8";
130 LPRINT TAB(58);"9     10     11     12"
140 LPRINT " ";STRING$(79,"-")
150 LPRINT " デンキ  |";:J=1
160 GOSUB 380
170 LPRINT " (kwh) |";
180 GOSUB 430
190 GOSUB 480
200 LPRINT " ガス   |";:J=2
210 GOSUB 380
220 LPRINT " (m3)  |";
230 GOSUB 430
240 GOSUB 480
250 LPRINT " スイドウ |";:J=3
260 GOSUB 380
270 LPRINT " (m3)  |";
280 GOSUB 430
290 LPRINT " ";STRING$(79,"-")
300 LPRINT
310 LPRINT TAB(15);"ゴウケイ       デンキ      ";
320 LPRINT USING "#### kwh";HYO(13,1)
330 LPRINT TAB(26);"ガス       ";
340 LPRINT USING "#### m3";HYO(13,2)
350 LPRINT TAB(26);"スイドウ    ";
360 LPRINT USING "#### m3";HYO(13,3)
370 END
380 '---------------ジッセキ インジ---------------
390 FOR I=1 TO 12
400 LPRINT USING " ### |";HYO(I,J);
410 NEXT I
420 RETURN
430 '---------------ヒリツ インジ-----------------
440 FOR I=1 TO 12
450 LPRINT USING "(###)|";HYO(I,J)/HYO(13,J)*100;
460 NEXT I:LPRINT
470 RETURN
480 '---------------ケイセン インジ---------------
490 LPRINT " ------+";
500 FOR I=1 TO 12
510 LPRINT "-----+";
520 NEXT I:LPRINT
530 RETURN
```

メインルーチン中のGOSUB文によって，サブルーチンに実行が移ります．また，各サブルーチンの最後には「RETURN（リターン）」が定義されていますが，この命令がみつかると，サブルーチンに実行を移したGOSUB文の次の文に実行を移します．つまり，GOSUB文と，それによって実行されるサブルーチンが置換されたように動作するわけです．

それではプログラムを分けて見ていきましょう．

```
10 DATA 350,370,330,360,345,352,368,375,351,340,320,380
20 DATA 68,70,63,58,57,55,53,59,60,58,62,67
30 DATA 63,57,58,60,61,53,57,65,60,58,57,60
40 DIM HYO(13,3)
50 FOR J=1 TO 3
60  FOR I=1 TO 12
70  READ HYO(I,J)
80  HYO(13,J)=HYO(13,J)+HYO(I,J)
90 NEXT I,J
```

ここでは，40行目のDIM文で確保された「HYO」という名前の配列変数に，10行～30行で定義されている実績データを読み込んでいます．読み込みの動作は，50行～90行までの２つのFOR～NEXT 文（FOR～NEXTの繰り返し）によって行っています．

DATA　350, 370, 330, ～, 320, 380
DATA　68, 70, 63, 58, ～, 62, 67
DATA　63, 57, 58, 60, ～, 57, 60

40行により確保された配列変数

J＼I	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1				3 3 0										
2														
3														

電気用 ガス用 水道用 変数

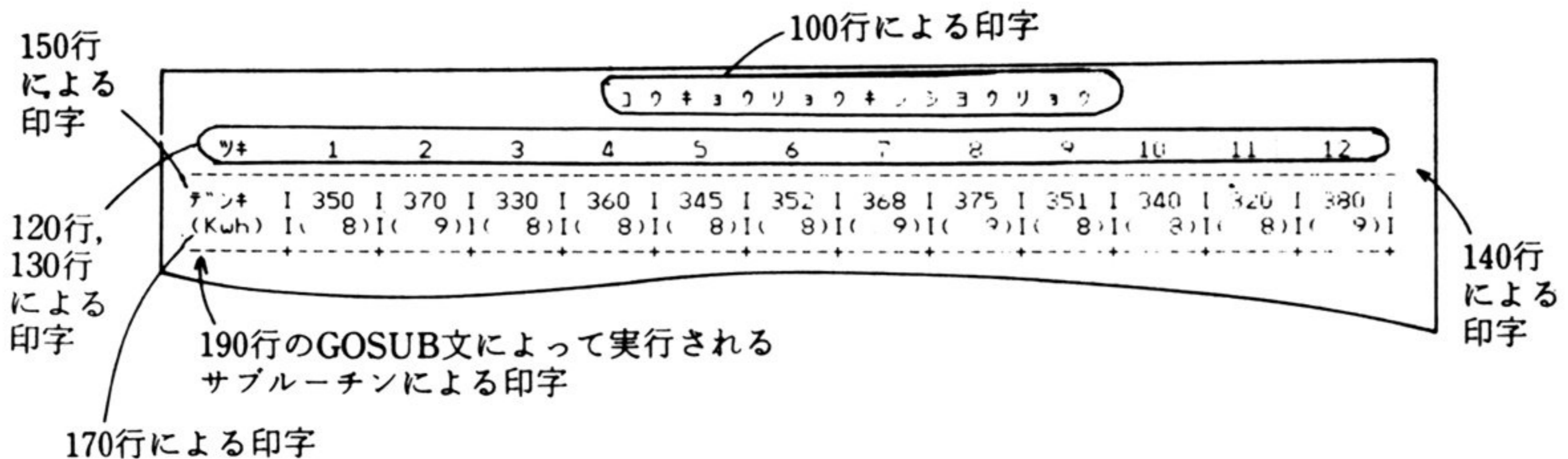
合計領域

I＝3，J＝1の時，「330」というデータが対応する領域に読み込まれる（HYO（3，1））

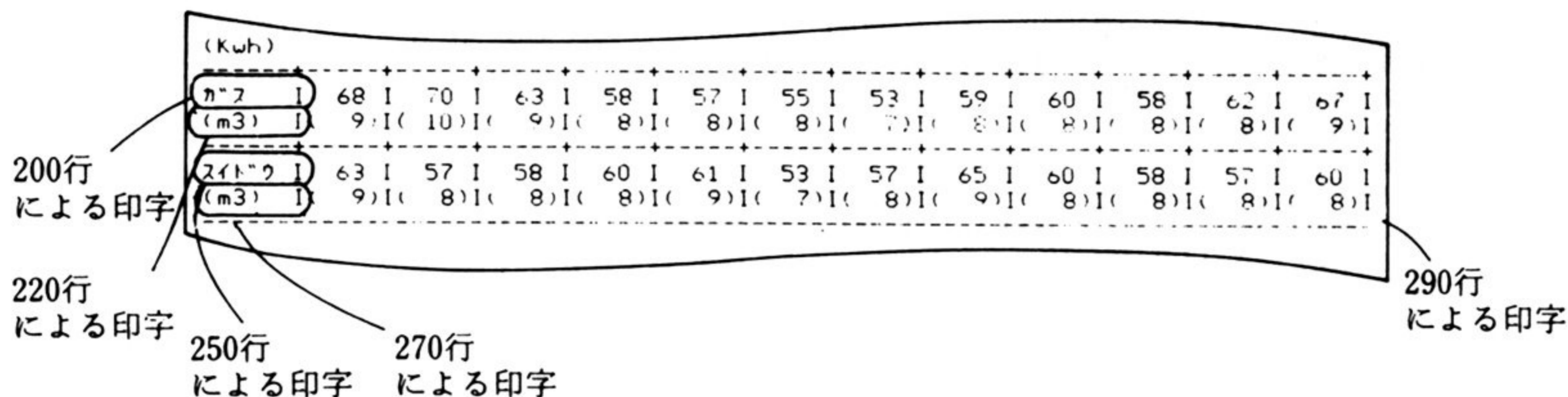
80行により各行の1～12までの内容が，13に加算される

▨は未使用領域

```
100 LPRINT TAB(29);"コ ウ キ ョ ウ リ ョ ウ キ ン シ ヨ ウ リ ョ ウ"
110 LPRINT
120 LPRINT "   ツキ     1      2      3      4      5      6      7      8";
130 LPRINT TAB(58);"9    10     11     12"
140 LPRINT " ";STRING$(79,"-")
150 LPRINT " デンキ  I";:J=1
160 GOSUB 380
170 LPRINT " (kwh) I";
180 GOSUB 430
190 GOSUB 480
```

100行～140行までの命令によって，プリンタの４行目までの見出し等を印字しています．140行目のSTRING$（ストリング）関数を使って，ハイフン(—)によるケイ線を印字しています．160, 180, 190行のGOSUB文により，各サブルーチンの実行に移ります．この150行から180行までは，電気料金の項の印字を行っているわけです．190行では，その下のケイ線印字を行っています．

```
200 LPRINT " ガス   I";:J=2
210 GOSUB 380
220 LPRINT " (m3)  I";
230 GOSUB 430
240 GOSUB 480
250 LPRINT " スイドウ I";:J=3
260 GOSUB 380
270 LPRINT " (m3)  I";
280 GOSUB 430
290 LPRINT " ";STRING$(79,"-")
```

200行～240行，250行～280行は，150行～190行と同様にガス，水道料金の項の印字を行っています．それぞれの配列変数と対応づけるために200行でJに２，250行でJに３を設定してい

ます．290行は13行目のケイ線の印字です．

注意

290行で使われている「STRING$」は，次のような意味をもっています．

STRING$ (79，〝—″)

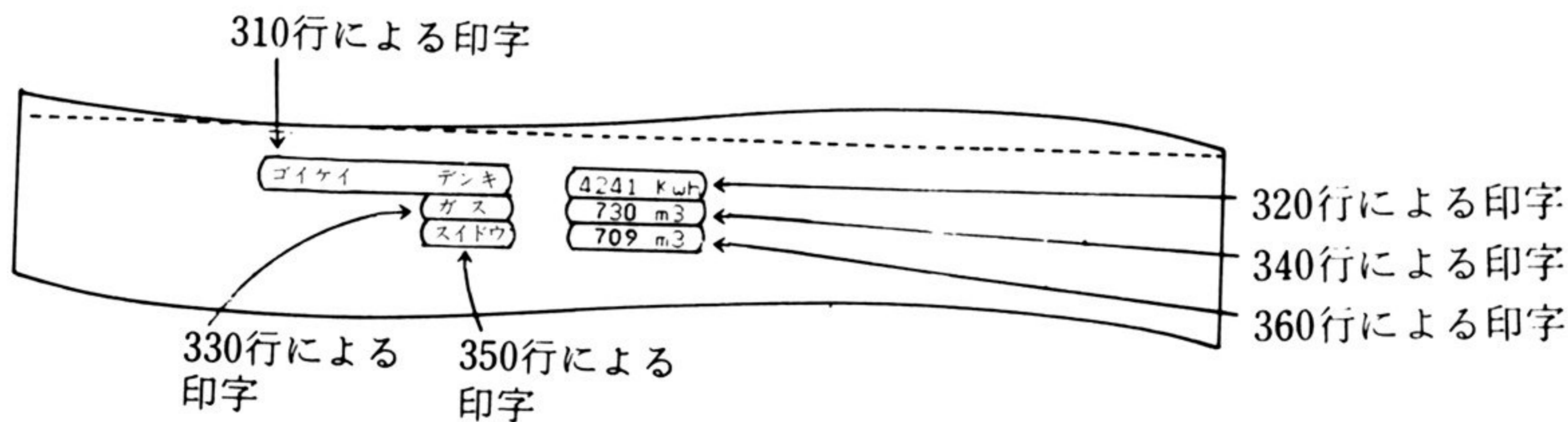
引用符（″）で囲まれた文字を，指定された数だけ与えます．

例では，ハイフン（—）を79桁与えていることになります．

つまり，LPRINT文の中で指定されていますから，ハイフンが79桁印字されることになります．

```
300 LPRINT
310 LPRINT TAB(15);"ゴウケイ        デンキ      ";
320 LPRINT USING "#### kwh";HYO(13,1)
330 LPRINT TAB(26);"ガス       ";
340 LPRINT USING "#### m3";HYO(13,2)
350 LPRINT TAB(26);"スイドウ     ";
360 LPRINT USING "#### m3";HYO(13,3)
370 END
```

ここでは合計の印字を行っています．

320行，340行，360行の「LPRINT USING（ユージング）」は，各命令の直後に引用符で囲んで指定されている書式に従って，セミコロン（；）以降のデータを印字します．

```
38Ø '----------------ｼﾞｯｾｷ ｲﾝｼﾞ----------------
39Ø FOR I=1 TO 12
4ØØ LPRINT USING " ### I";HYO(I,J);
41Ø NEXT I:LPRINT
42Ø RETURN
43Ø '----------------ﾋﾘﾂ ｲﾝｼﾞ-----------------
44Ø FOR I=1 TO 12
45Ø LPRINT USING "(###)I";HYO(I,J)/HYO(13,J)*1ØØ;
46Ø NEXT I:LPRINT
47Ø RETURN
48Ø '----------------ｹｲｾﾝ ｲﾝｼﾞ----------------
49Ø LPRINT " ------+";
5ØØ FOR I=1 TO 12
51Ø LPRINT "-----+";
52Ø NEXT I:LPRINT
53Ø RETURN
```

380～420行，430～470行，480～530行がメインルーチンからGOSUB文で参照するサブルーチンです．サブルーチンの最後にはRETURN文が必要です．

380～420行では，１月から12月の実績を印字する作業を行っています．

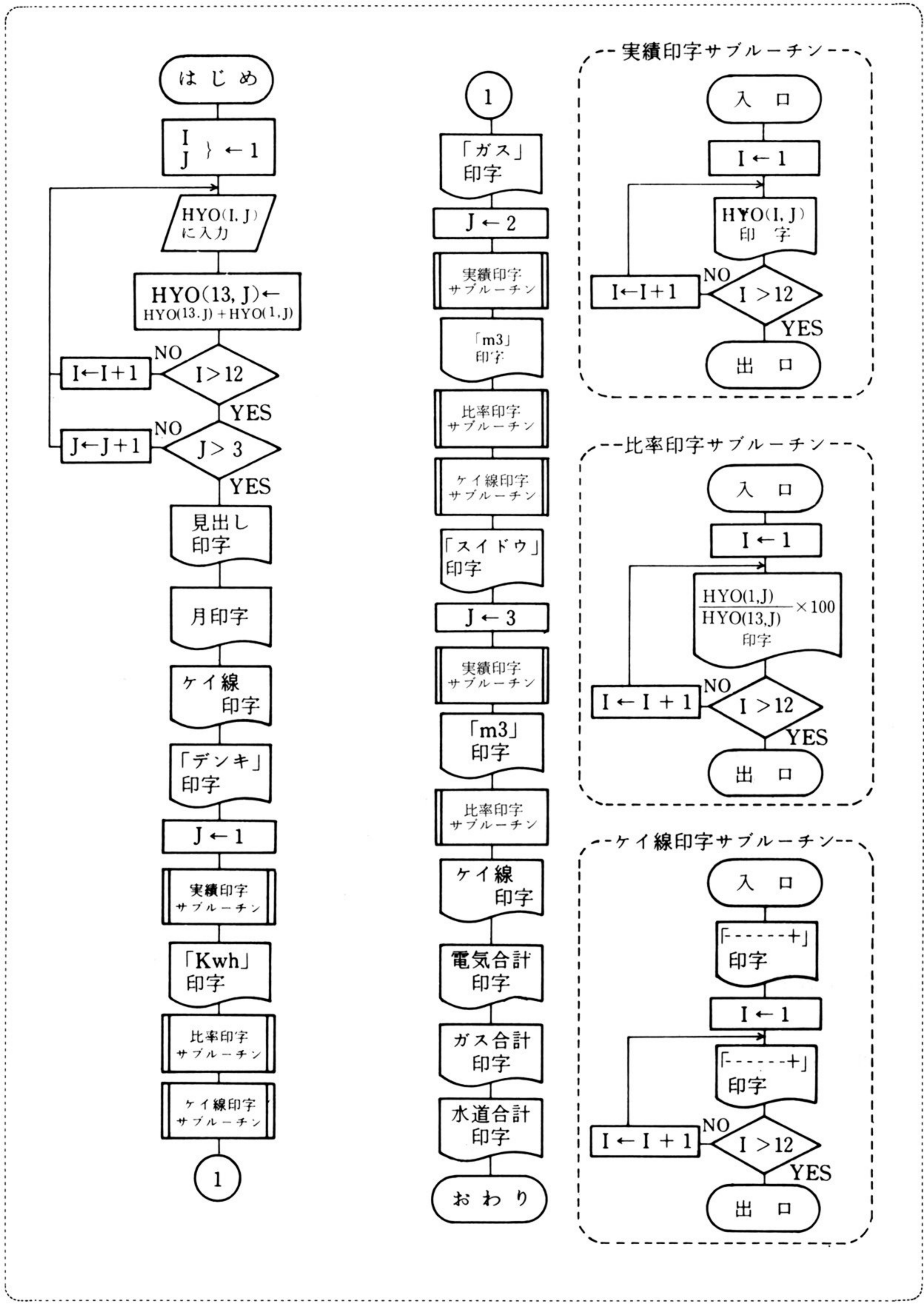
430～470行では，実績の下に各月の年間実績に対する比率を印字しています．

480～530行では，電気とガス，ガスと水道間のケイ線の印字を行っています．

各サブルーチンの最初にアポストロフィ(')で始まる文がありますが，これは注釈の行です．プログラムを実行するうえで何ら影響はありません．

このプログラムでは，各サブルーチンの意味を表す注釈が定義されています．

[3] 処理の手順

プログラム例をフローチャートで考えてみましょう．

このプログラムは，あらかじめ確保した配列にデータを入力し，各料金ごとの合計を計算する処理を先に行っています．そして，配列内の値を順番に入力しながらプリンタに印字しています．つまり，今までの「入力→処理→出力」の繰り返しのパターンとは少し異なっているわ

けです．

プログラムの基本型は以前説明したような形になっていますが，全てのプログラムがそのような形になっているということではなく，場合によっては今回のような特殊なパターンになるものもいくつかあるでしょう．

また，今回のプログラム例では「サブルーチン」の機能が頻繁に使用されています．

プログラム中で何度も使用する処理を1ケ所に定義しておき，それを必要に応じて参照するというのが「サブルーチン」の機能ですが，今回は実績印字を行うサブルーチン，比率印字のサブルーチン，ケイ線印字のサブルーチンを使っています．各処理は，FOR～NEXT文を使って12カ月分の繰り返しを行っています．これを，電気，ガス，水道ごとに実行しますから，そのたびに，これらのサブルーチンが参照されることになります．

この様に，サブルーチン機能を利用することにより，プログラムを簡潔にすることができますから，同一処理が何カ所もでてくるようなプログラムはサブルーチンを作るようにします．

注意

実績値，比率を印字する際にHYO（I,J）やHYO（13，J）などが使われていますが，括弧内の数字は「添え字」を表わしています．

あらかじめ確保された，実績値や合計を入れるための配列変数の特定の変数を参照するために使っている識別子です．

HYO（13,J）の中，つまり，HYO（13,1），HYO（13,2），HYO（13,3）にはそれぞれ電気の合計，ガスの合計，水道の合計が入っていますので，HYO(I,J)のIを1～12，Jを1～3まで変えて，それぞれの合計で割ることによって，各月の合計値に対する比率を求めることができます．I，Jの変更はFOR～NEXT文で簡単にできますね．

このように，関連する多くの変数を配列変数として確保することにより，添え字をFOR～NEXT文の制御変数として使うことができるわけです．プログラムの流れがわかり易くなることが明確です．

4 命令の説明

本章で新しく出てきた命令を，見ていくことにしましょう．

① DIM（ディム）文

配列変数を確保します．

書き方

DIM　変数名（添字の最大値〔，添字の最大値〕…）

DIM文を書く時の注意は次のとおりです．

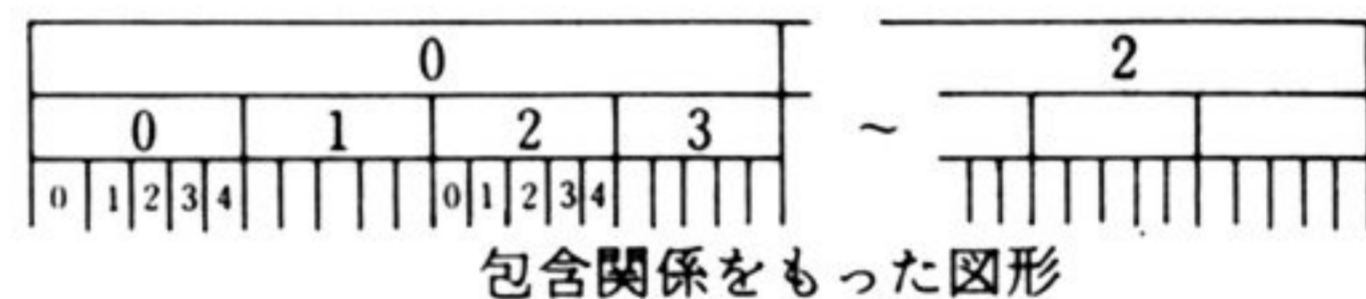
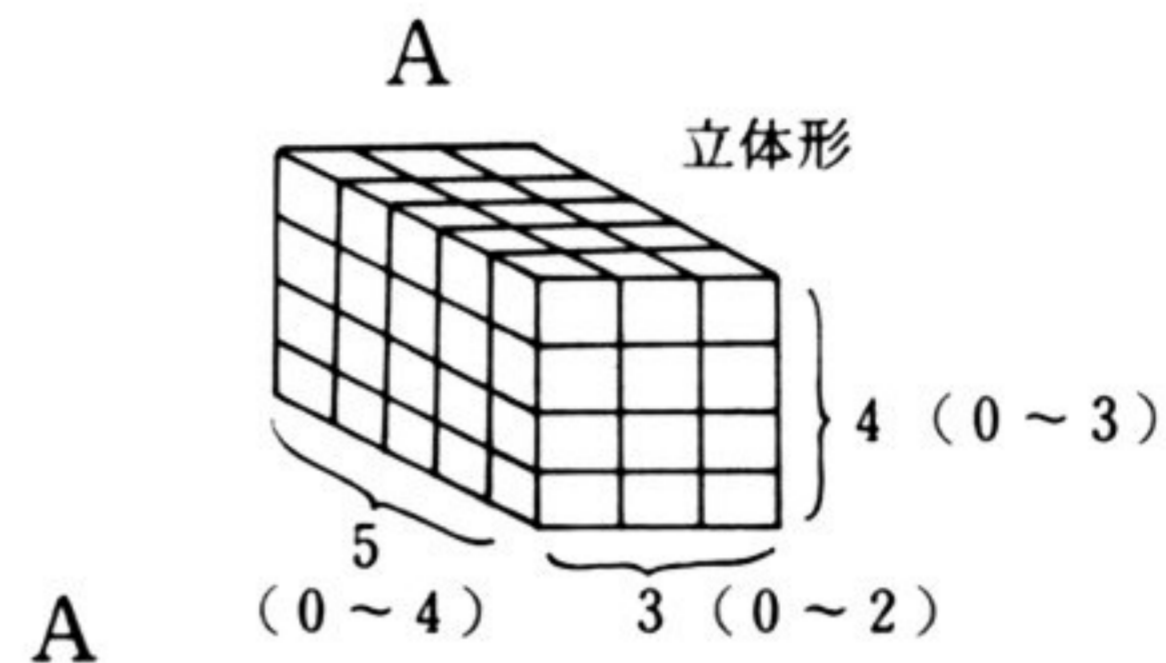
・変数名は，配列変数全体につけられた名前です.
・添字の最大値は，コンマで区切って255個まで指定できます.
・通常は，添字は０番からつけられます．ただし，これは，OPTION BASE文で１番からに変更することができます．詳細は，リファレンスマニュアルを参照して下さい.
・DIM文で確保された変数は，数値は０に，文字は空白になっています.

例

(1) DIM A（2，3，4）

このような指定を，３次元の配列と呼びます．つまり，添え字の最大値の個数が，次元数になります．右図のように，３次元以上の表においては，どのような形をもった図形を想像してもかまいません.

皆さんが使おうとしているイメージで形を考えていただければよいわけです.

また，この配列変数には「A」という名前がついていますから，単精度の変数であることがわかります．このDIMが実行されると，上図のような変数が確保されるとともに，各変数には「０」が入ります.

② GOSUB（ゴーサブ）文

指定されたサブルーチンに，実行の流れを移します.

書き方

GOSUB　行番号

・行番号はサブルーチンの最初の行番号でなければなりません.

③ RETURN（リターン）文

サブルーチンの実行を終え，サブルーチンを呼び出したGOSUB文の次の文へ流れを移します.

書き方

RETURN　〔行番号〕

・行番号を指定すると，指定された行番号の行へ実行を戻します.

④ LPRINT USING文

データを指定した書式で印字します．

書き方

LPRINT USING〝書式制御文字列″；式〔{，；} 式〕…

LPRINT USING文を書く時の注意は次のとおりです．

・書式制御文字列では，プリンタに印字するデータの形式を指定します．
・式と式の区切りに使われるセミコロン（；）とコンマ（,）はどちらを使っても違いはありません．
・書式制御文字としては，いろいろなものが用意されていますが，詳細はリファレンスマニュアルを参照して下さい．
・書式制御文字以外の文字を書式制御文字列で指定すると，その文字がそのまま，その位置に印字されます．

例

(1)　LPRINT USING〝△△###．##″；123，23.5，3.856

①は，数字がそのまま印字されます．ただし書式の中の「.」は小数点の位置を表しますから，小数点の位置が桁合わせして印字されています．また小数部は，必ず「#」の数だけ印字されます．

②は，同じく小数点で桁合わせして印字されていますが，整数部が2桁しかないため，余った「#」の桁は空白に置き換わります．

③は，小数部の指定が2桁（～，##）しかないのに，データの方は小数部が3桁あります．このような場合は，指定された小数部の桁（例では2桁）の次の桁を四捨五入し指定された桁数分だけを印字します．

(2)　LPRINT USING〝タンカ△△#######，△エン″；1234567

印字すると

タンカ△△1,234,567△エン

この場合は「#」の前後の「タンカ」と「エン」がそのまま，指定された位置に印字されます．また「#」の文字列の最後に「,」がありますが，この指定をすると，データを3桁ごとにコンマで区切って印字します．

⑤　STRING$（ストリング）関数

任意の文字を任意の数だけ与えます．

書き方

STRING$（式{文字式／数　式}）

STRING$関数を使う時の注意は，次のとおりです．

・文字式を指定すると，最初の1文字が有効になります．
・数式を指定すると，その数値をキャラクタコードとみなし，対応する文字を取り扱います．
・式は，指定された文字の必要桁数を指定します．

⑥ REM（レム）文

プログラム中に注釈を入れます．

書き方

REM〔注釈文〕

REM文を書く時の注意は，次のとおりです．

・REM文の行は，すべてが注釈になります．
・「REM」の代わりにアポストロフィ（'）を使うことができます．

第5章　ファイルの問い合わせ

本章では，任意のデータを直接入力したり，直接出力したりすることができるファイルの作り方と，そのファイルを使った問い合わせのプログラムについて学習します．

本章のポイント

- ランダムファイルの作成
- ランダムファイルを使った問い合わせの方法

第１節　ランダムな入出力ができるファイルの作成

ここでは，任意のデータを直接入力したり，出力したりすることができる「ランダムファイル」の作成方法を学びます．

⑴　住所録ファイルの作成（例題５－１）

例題の概要

次のような画面のもとで，住所録データのフロッピィディスクへの登録を行います．

◎住所録データの登録は，登録番号をもとに行います．

◎登録番号は，次のような規則のもとに番号付けを行います．

×××　　　　氏名の頭文字の種類により
↑　↑　　　　区分番号を付け分けます．
50音別区分　連番

50音別区分は，次のように規定しておきます．

50　音	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ
区分番号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
50　音	ハ	ヒ	フ	ヘ	ホ	マ	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン					
区分番号	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44					

ここでは，同一50音で始まる氏名は10件以内ということに仮定しておきましょう．つまり，連番は１桁（０～９）の範囲です．

◎氏名は13桁，住所は40桁以内の入力とします．

◎郵便番号は５桁で入力します．ただし，３桁でよいものは３桁で登録します．

◎電話番号は，市外局番，市内局番，加入者番号の間にハイフン（－）を入れます．

処理の条件

・1人分のデータ登録後，19行目で「次の処理」として何を行うかの確認入力を行います．確認入力の種類とそれに応じた処理は次のとおりです．

入力文字

C……………今入力されたデータをフロッピィに出力し，次のデータを入力できるよう画面を表示し直します．

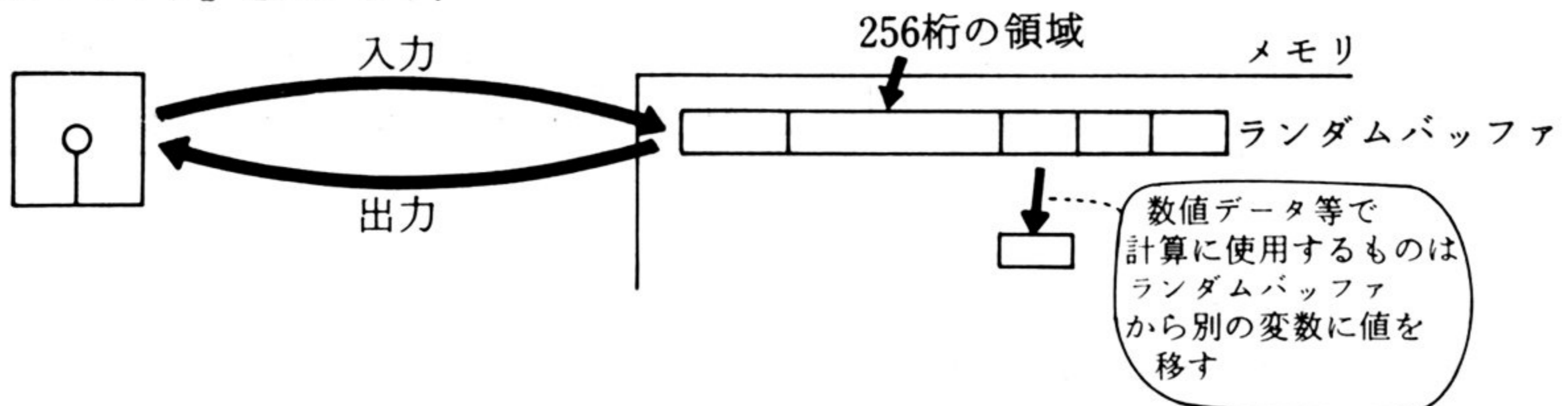
B……………今入力したデータに入れ間違いがあった場合で，登録番号の位置へ戻し，入れ直しが可能な状態にします．

E……………今入力されたデータをフロッピィに出力し，プログラムを終了します．

プログラム作成上の注意点

いままで取扱ってきたファイルは，すべてシーケンシャルファイルと呼ばれるファイルでしたが，今回取り扱う「ランダムファイル」は，次のような点でシーケンシャルファイルと異なっています．

(1) ファイル準備のOPEN文で，「FORモード」は省略します．

(2) フロッピィディスク上のファイルとメモリ間でのデータのやりとりは，必ず「ランダムバッファ」を用います．

なお，ランダムバッファはOPENが実行された時点で，自動的に確保されます．

(3) ランダムファイルからのデータの入力には「GET」文，ランダムファイルへのデータの出力には「PUT」文を使います．

2 プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．

```
10 OPEN "1:JUSYOF" AS #1
20 FIELD 1,13 AS SHIMEI$,40 AS JUSYO$,6 AS UBIN$,13 AS TELNO$
30 PRINT CHR$(12)
40 LOCATE 26, 1:PRINT "ｼﾞ ｭ ｳ ｼ ｮ ﾛ ｸ  ｻ ｸ ｾ ｲ"
50 LOCATE  9, 3:INPUT "ﾄｳﾛｸﾊﾞﾝｺﾞｳ ----->  ";BANGO
60 LOCATE  9, 6:INPUT "ｼ  ﾒ  ｲ  ------->  ";KSHIMEI$
70 LOCATE  9, 9:INPUT "ｼﾞｭｳｼｮ  --------->  ";KJUSHO$
80 LOCATE  9,12:INPUT "ｺｳﾋﾞﾝﾊﾞﾝｺﾞｳ ---->  ";KUBIN$
90 LOCATE  9,15:INPUT "TEL-NO  --------->  ";KTELNO$
100 LOCATE 53,19:INPUT "NEXT PROCESS (C/B/E) ";NP$
110 IF NP$="C" THEN GOSUB 160:GOTO 30
120 IF NP$="B" THEN 50
130 IF NP$="E" THEN GOSUB 160:CLOSE:END
140 GOTO 100
150 '-------------FDD PRINT---------------
160 LSET SHIMEI$=KSHIMEI$:LSET JUSHO$=KJUSHO$
170 LSET UBIN$=KUBIN$:LSET TELNO$=KTELNO$
180 PUT 1,BANGO
190 RETURN
```

150行から190行は，キーボードより入力したデータをフロッピィディスクに書き出すサブルーチンです．

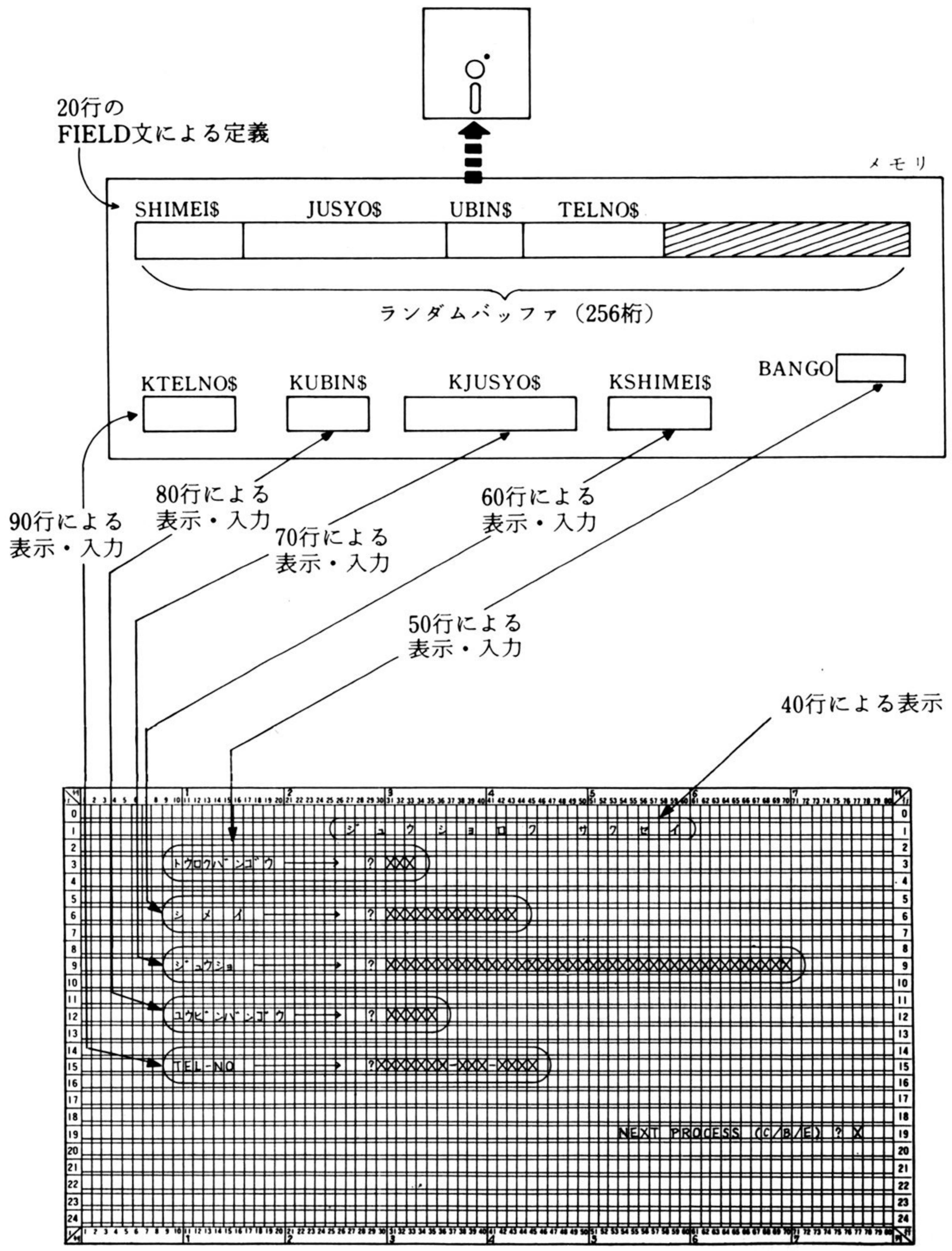
それではプログラムを分けて見ていきましょう。

```
10 OPEN "1:JUSHOF" AS #1
20 FIELD 1,13 AS SHIMEI$,40 AS JUSHO$,6 AS UBIN$,13 AS TELNO$
30 PRINT CHR$(12)
40 LOCATE 26, 1:PRINT "ｼﾞ ｭ ｳ ｼ ｮ ﾛ ｸ  ｻ ｸ ｾ ｲ"
50 LOCATE  9, 3:INPUT "ﾄｳﾛｸﾊﾞﾝｺﾞｳ ----->  ";BANGO
60 LOCATE  9, 6:INPUT "ｼ  ﾒ  ｲ  ------->  ";KSHIMEI$
70 LOCATE  9, 9:INPUT "ｼﾞｭｳｼｮ  --------->  ";KJUSHO$
80 LOCATE  9,12:INPUT "ｺｳﾋﾞﾝﾊﾞﾝｺﾞｳ ---->  ";KUBIN$
90 LOCATE  9,15:INPUT "TEL-NO  --------->  ";KTELNO$
```

10行目は，ファイル準備の命令ですね．ランダムファイルですから，「FORモード」が省略されています．また，この命令によって，ランダムバッファ(256桁)がメモリ上に確保されます．

20行目は，「FIELD（フィールド）」文によって，ランダムバッファを項目ごとに区切っています．たとえば256桁の先頭から13桁に「SHIMEI$」という名前をつけて，氏名を入れる領域を割り当てているわけです．

30行目では，画面を消去し，60行～90行で「氏名」，「住所」，「郵便番号」，「電話番号」等の住所録データを，キーボードから入力しています．各変数の名前が，ランダムバッファで定義されている変数名とは違っていることに注意して下さい．これはキーボードから入力したデータを，直接ランダムバッファの各項目へ入れることはできないからです．

また，50行で入力している「BANGO」は，これから登録するデータを何番目のデータとして登録するかを，決める番号です．この番号が，特定のデータを入力したり出力する時の呼び出し番号になります．

```
100 LOCATE 53,19:INPUT "NEXT PROCESS (C/B/E) ";NP$
110 IF NP$="C" THEN GOSUB 160:GOTO 30
120 IF NP$="B" THEN 50
130 IF NP$="E" THEN GOSUB 160:CLOSE:END
140 GOTO 100
150 '-------------FDD PRINT---------------
160 LSET SHIMEI$=KSHIMEI$:LSET JUSHO$=KJUSHO$
170 LSET UBIN$=KUBIN$:LSET TELNO$=KTELNO$
180 PUT 1,BANGO
190 RETURN
```

100行では，次の処理として何を行うかを示す確認入力として文字を入れさせています．入

力された文字により以降で行う作業が異なりますから，110～140行で，IF文で押された文字を調べ，それによって処理をふり分けています．

「C」が押されたならば，160行から始まるサブルーチンを実行し，フロッピィディスクにデータを出力した後，30行に実行を移しています．別のデータの入力を続けることになります．

「B」が押されたならば，入力ミスがあったということで，50行へ戻し，登録番号から入れ直しを行わせます．

「E」が押された場合は，まず入力済みのデータをフロッピィディスクへ出力し，プログラムを終了にしています．

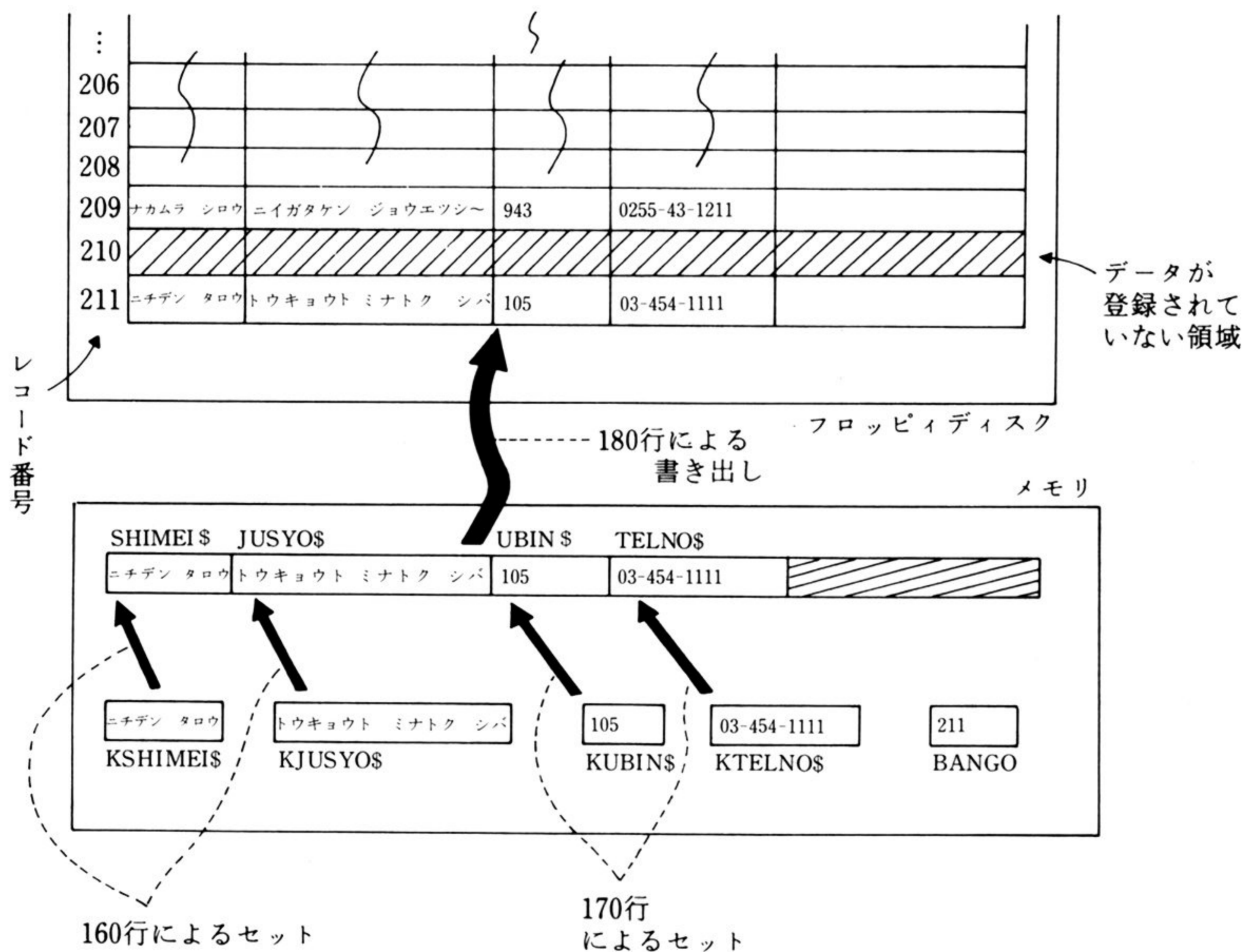
140行では上記以外のキーが押された時に100行へ戻し，正しいキーを入れ直させています．

150行からのサブルーチンで，フロッピィにデータを書き出しています．

160行から170行にかけて使われている「LSET（エルセット）」文は，20行で定義されているランダムバッファの各変数に，キーボードより入力された値をセットする命令です．「LSET」の「L」はLeft（左）の意味で，ランダムバッファの各変数に値をセットする際，左側につめてセットします．

そして，データをすべてランダムバッファにセットした後で，180行の「PUT」文により，ランダムバッファの内容をフロッピィに書き出します．

「PUT」の後の「1」は，ファイル番号です．10行目のOPEN文のファイル番号と対応づけられています．また，その後にコンマで区切り，「BANGO」とありますが，ここで指定された番号が，データの登録番号になります．この登録番号のことを「レコード番号」と呼びます．

190行目の「RETURN」はサブルーチンの出口を表す命令です．

注意

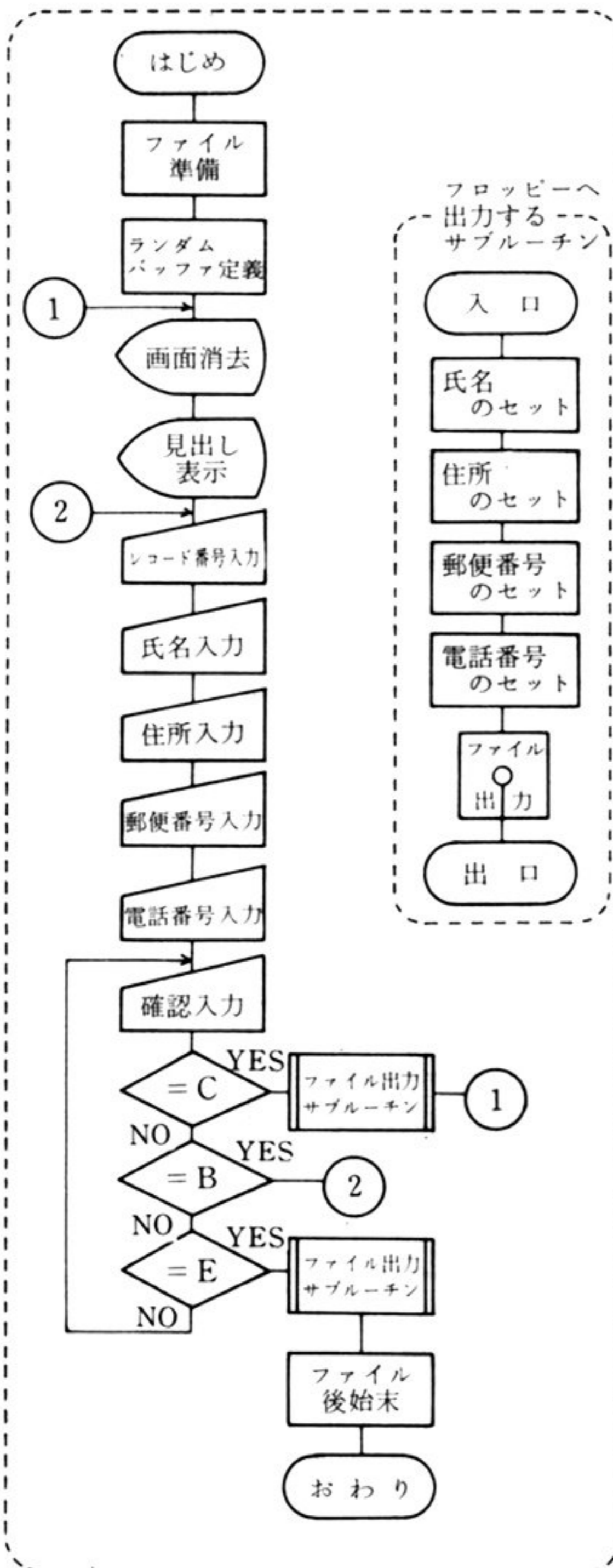
(1) 前項の図でも出てきていますが，1件のデータのことを「レコード」と呼び，ファイル中の他のデータとの相対的な位置関係を表す番号を,「レコード番号」と呼んでいます．

(2) ランダムファイルの場合，任意の位置へデータを書き出すことができますから，途中にデータが記録されていない領域が存在する場合があります．

(3) 既にデータが登録されているレコード番号の位置へ，新たにデータを書き出すと，以前のデータはこわされ，新しいデータが記録されます．

(4) 数値データ（整数変数，単精度変数，倍精度変数に入っているデータ）は，直接ランダムバッファにセットすることはできません．MKI$，MKS$，MKD$等の関数を使って形を変換してからセットしなければなりませんし，詳細はリファレンスマニュアルを参照して下さい．

③ 処理の手順

プログラム例を，フローチャートで考えてみましょう．

フローチャートの流れを見ても分かるように，ランダムファイルを取り扱うプログラムは，今までのシーケンシャルファイルを取り扱っているプログラムと何ら変わりはありません．キーボードから入力したデータを，ランダムバッファにセットし(処理)，それをフロッピィに書き出しているわけです（出力）．

違いは，ランダムファイルを使用する場合に必須のランダムバッファの定義が必要なことです．ランダムバッファの定義とは「OPEN」命令で確保された256桁の領域を項目ごとに分けて「FIELD文」で区切ることです．また，フロッピィへのデータの書き出しは，ランダムバッファの単位（256桁）で行います．

なお，データの出力の際は，任意の位置への書き出しが可能ですから，プログラム中で，どの位置に書き込むかその位置を示すレコード番号を入力させる処理も必要になってくるでしょう．

また，このプログラムにおいて，次の処理入力で「B」が押された時，データの入れ直しが可能になっています．しかしキーボードを操作していくうえで，操作ミスというものは避けることはできません．そこで，プログラム中に操作ミスが発生した時の修正のための処理を入れておくことにより，プログラムをより完全なものにすることができる訳です．

④ 命令の説明

本節で新しく出てきた命令を，見ていくことにしましょう．

① FIELD文

ランダムバッファをいくつかの変数領域に区切ります.

書き方

FIELD〔#〕ファイル番号，フィールド幅 AS 文字変数名
〔，フィールド幅 AS 文字変数名〕…

FIELD文を書く時の注意は，次のとおりです.

・ファイル番号は，OPEN文によりファイルを準備した時に指定した番号です.
・フィールド幅は，「AS」で対応づけられた変数に割り当てられる領域の桁数です. 1つの変数は最大で255桁まで割り当てできます. また，FIELD文で指定するフィールド幅の合計は256桁以内でなければなりません.
・この文は，GET文，PUT文の前に指定されていなければなりません

②LSET／RSET文

ランダムバッファの各変数領域に値をセットします.

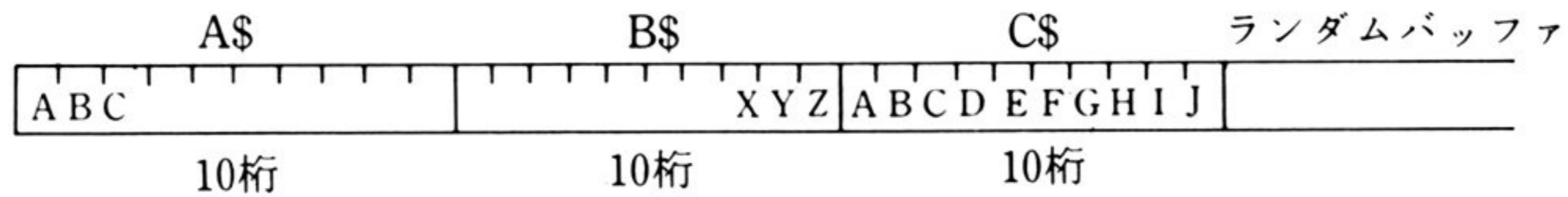
書き方

{LSET / RSET} 文字変数名＝文字データ

LSET／RSET文を書く時の注意は次のとおりです.

・ランダムバッファにデータをセットする際は，必ずLSET／RSETを使わなければなりません.
・LSET／RSETを使ってランダムバッファにセットするデータは，文字データでなければなりません. 数値データは，「MKI＄」，「MKS＄」，「MKD＄」のいずれかで文字データに変換しておく必要があります.
・文字データの桁数よりも，対応するフィールド幅の方が長い場合は，LSETが左詰め，RSETが右詰めにデータをセットします.
・文字データの桁数がフィールド幅より長い場合は，余分な右側の桁が切り捨てられます.

例

```
10  OPEN "1:SAMPLE" AS 1
20  FIELD 1,10 AS A$,10 AS B$,10 AS C$
30  LSET A$= "ABC" : RSET B$= "XYZ" : RSET C$= "ABCDEFGHIJKL"
40  PUT 1
50  CLOSE:END
```

③ PUT文

ランダムファイルへランダムバッファの内容を出力します。

書き方

PUT〔#〕ファイル番号〔, レコード番号〕

PUT文を書く時の注意は,次のとおりです.

・指定されたファイル番号に対応するファイルにランダムバッファの内容を，書き出します.
・レコード番号が指定された場合は，その番号に対応する位置へ，省略された場合は，順番にデータを出力していきます.

例

第2節　ランダムファイルを使った問い合わせ

ここでは，第1節で作成したランダムファイルの問い合わせを行うプログラムを考えてみましょう。プログラム作成上の注意は第1節と同様です。

1　住所録ファイルの問い合わせ（例題5－2）

例題の概要

次のような画面を使って，登録番号をもとに住所録ファイルの問い合わせを行います．

処理の条件

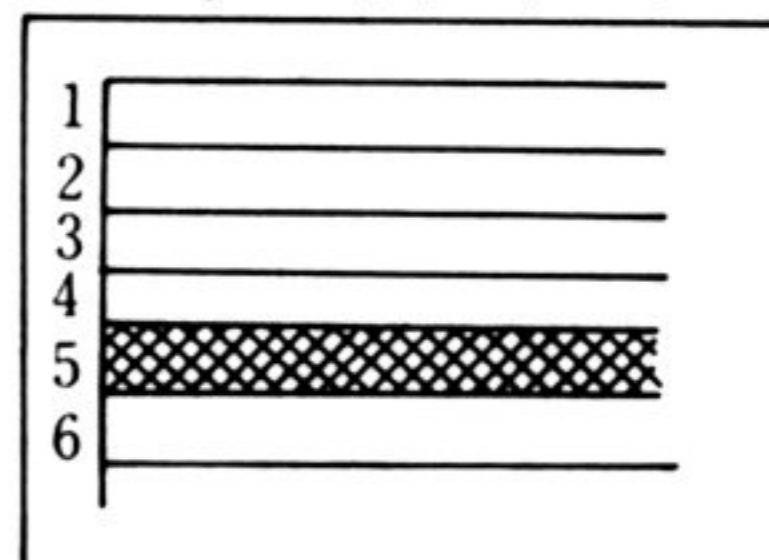
・画面3行目で，問い合わせを行いたいデータの登録番号（レコード番号）を入力し，該当するデータをファイルより入力します．

・各項目の表示位置は上図のとおりですが，郵便番号の前3桁と後2桁の間にはハイフン（－）を表示します．

・1件分のデータ表示後，画面19行目において次の処理を示す確認入力を行います．入力文字の種類とそれに応じた処理は次のとおりです．

入力文字

C …………画面を表示し直し，別データの問い合わせができるようにします．

E …………プログラムを終了させます．

2 プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．

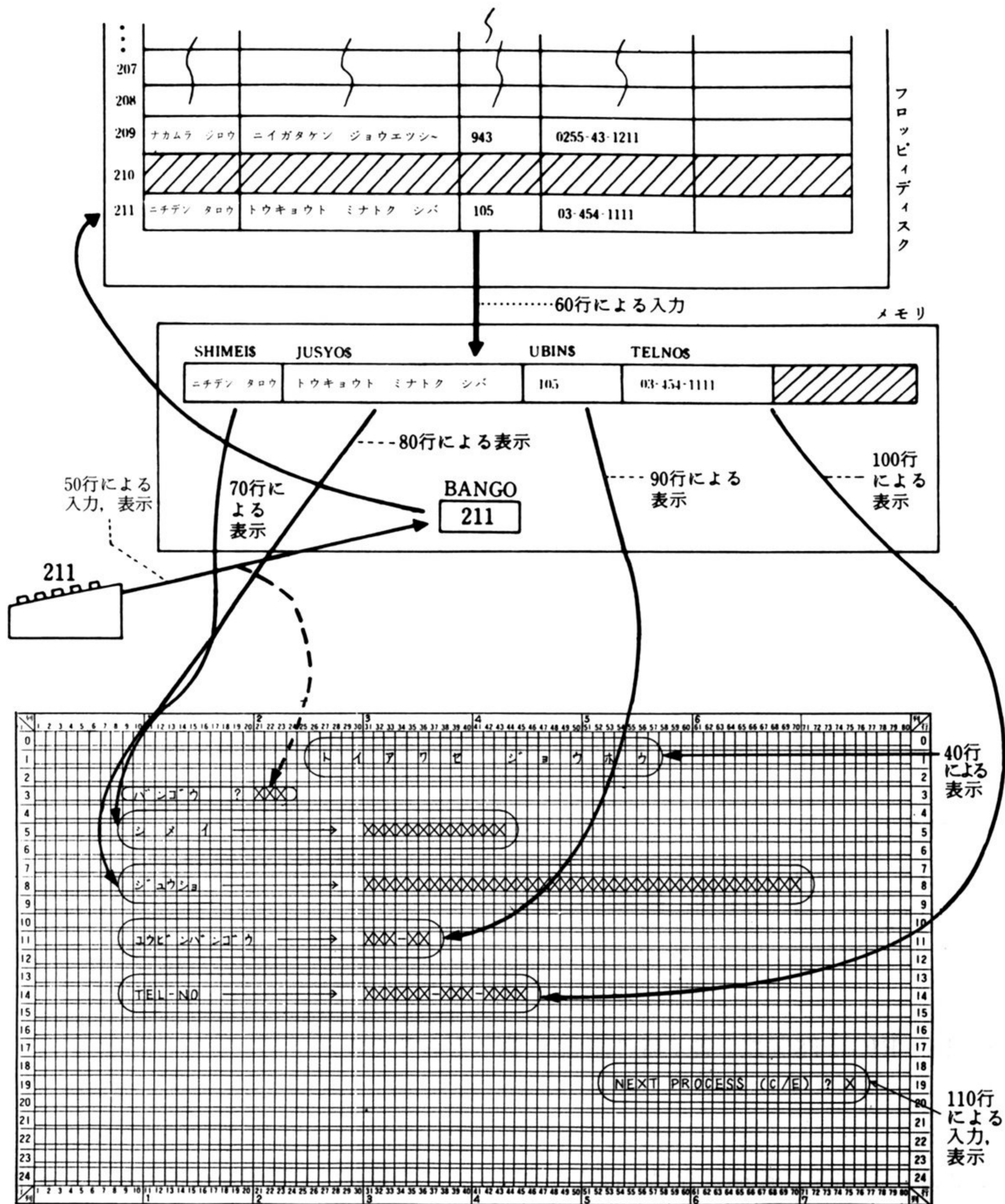
```
10 OPEN "JUSYOF" AS #1
20 FIELD 1,13 AS SHIMEI$,40 AS JUSYO$,5 AS UBIN$,13 AS TELNO$
30 PRINT CHR$(12)
40 LOCATE 26, 1:PRINT "ﾄ ｲ ｱ ﾜ ｾ  ｼﾞ ｮ ｳ ﾎ ｳ"
50 LOCATE 18, 3:INPUT "ﾊﾞﾝｺﾞｳ   ";BANGO
60 GET 1,BANGO
70 LOCATE  9, 5:PRINT "ｼ ﾒ ｲ --------->  ";SHIMEI$
80 LOCATE  9, 8:PRINT "ｼﾞｭｳｼｮ ---------->  ";JUSYO$
90 LOCATE  9,11:PRINT "ﾕｳﾋﾞﾝﾊﾞﾝｺﾞｳ ----->  ";LEFT$(UBIN$,3);"-";RIGHT$(UBIN$,2)
100 LOCATE 9,14:PRINT "TEL-NO ---------->  ";TELNO$
110 LOCATE 53,19:INPUT "NEXT PROCESS (C/E) ";NP$
120 IF NP$="C" THEN 30
130 IF NP$="E" THEN 140 ELSE 110
140 CLOSE:END
```

10行目でファイルを準備し，20行目でランダムバッファの区分けを行っています．30行，40行は，画面消去と見出し表示です．

50行目で，変数BANGOにレコード番号が入力されますから，60行目の「GET（ゲット）」文で，指定されたレコード番号の位置からデータを入力してきます．

入力されたデータを各項目ごとに所定の位置へ表示しているのが，70～100行の命令です。

表示内容を確認後，110行目で「次の処理」に対する確認入力を行います．ここで押されたキーが何であるかを120行，130行で調べ，その文字に応じて処理を行っています．「C」が押されていれば，30行に戻り別のデータの問い合わせを行います．「E」が押されていれば，プログラムの終了にします．また押されたキーが，「C」でもなく「E」でもない時は110行へ戻り，キーを入れ直させます．

90行目に新たに出てきたLEFT（レフト）$，RIGHT（ライト）$関数は，文字の並び（文字列）の左または右側から指定文字数だけ抽出してくる命令です．郵便番号は5桁の文字で登録されているので，3桁と2桁に分割して取り扱うためには，これらの関数を使う必要があります．

③ 命令の説明

本節で新しく出てきた命令を，見ていくことにしましょう．

① GET(ゲット)文

ランダムファイルの中のデータを，ランダムバッファに読み込みます．

書き方

GET〔#〕ファイル番号〔，レコード番号〕

GET文を書く時の注意は，次のとおりです．

・指定されたファイル番号に対応するファイルからランダムバッファに，データを読み込みます．
・レコード番号が指定された場合は，その番号に対応する位置から，省略された場合は，以前に入力したデータの次のデータが入力されてきます．

② LEFT$関数／RIGHT$関数

指定された文字列の左側から，または右側から指定された桁数を抽出します．

LEFT$ RIGHT$関数を書く時の注意は，次のとおりです．

・数式は，0～255の範囲になければなりません．この値が，文字列から抽出してくる桁数になります．
・数式の値が，文字列の桁数を超えている場合は，文字列をすべて取り出します．
・数式が0の時は，何もとり出しません．

第6章　グラフを作ってみよう

5章までは，各種のデータを表に表わしたり，ファイルに出力したりしました。本章では，データをグラフに表わすことに挑戦してみましょう．

本章のポイント

- 画面上に任意の線や四角を表示する方法
- 画面上に円，および，扇形を表示する方法
- グラフに色を塗る方法

第1節　棒グラフと円グラフの作成

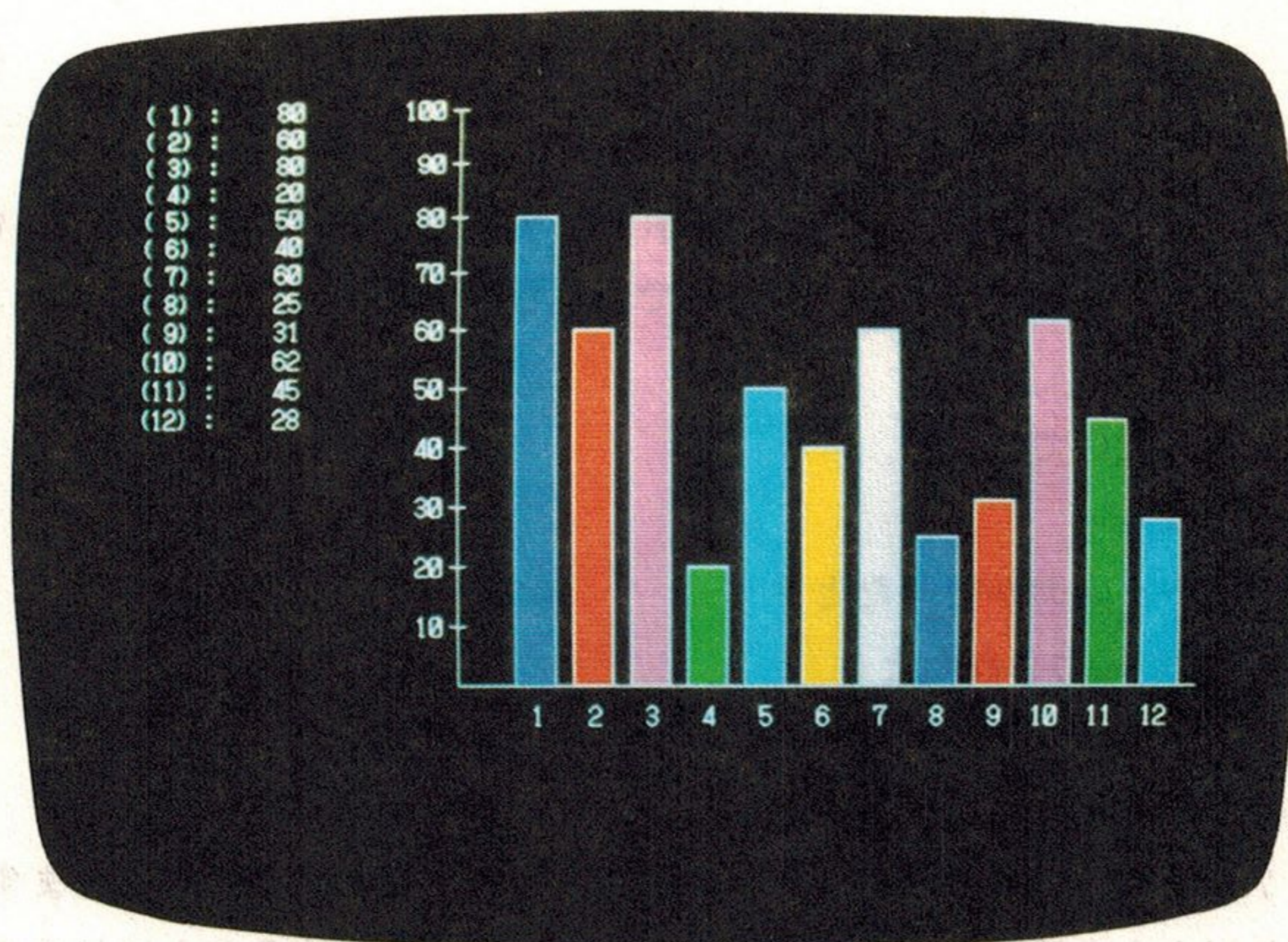
ここでは，棒グラフと円グラフの表示方法を学びます．

1　棒グラフと円グラフの作成（例題6－1）

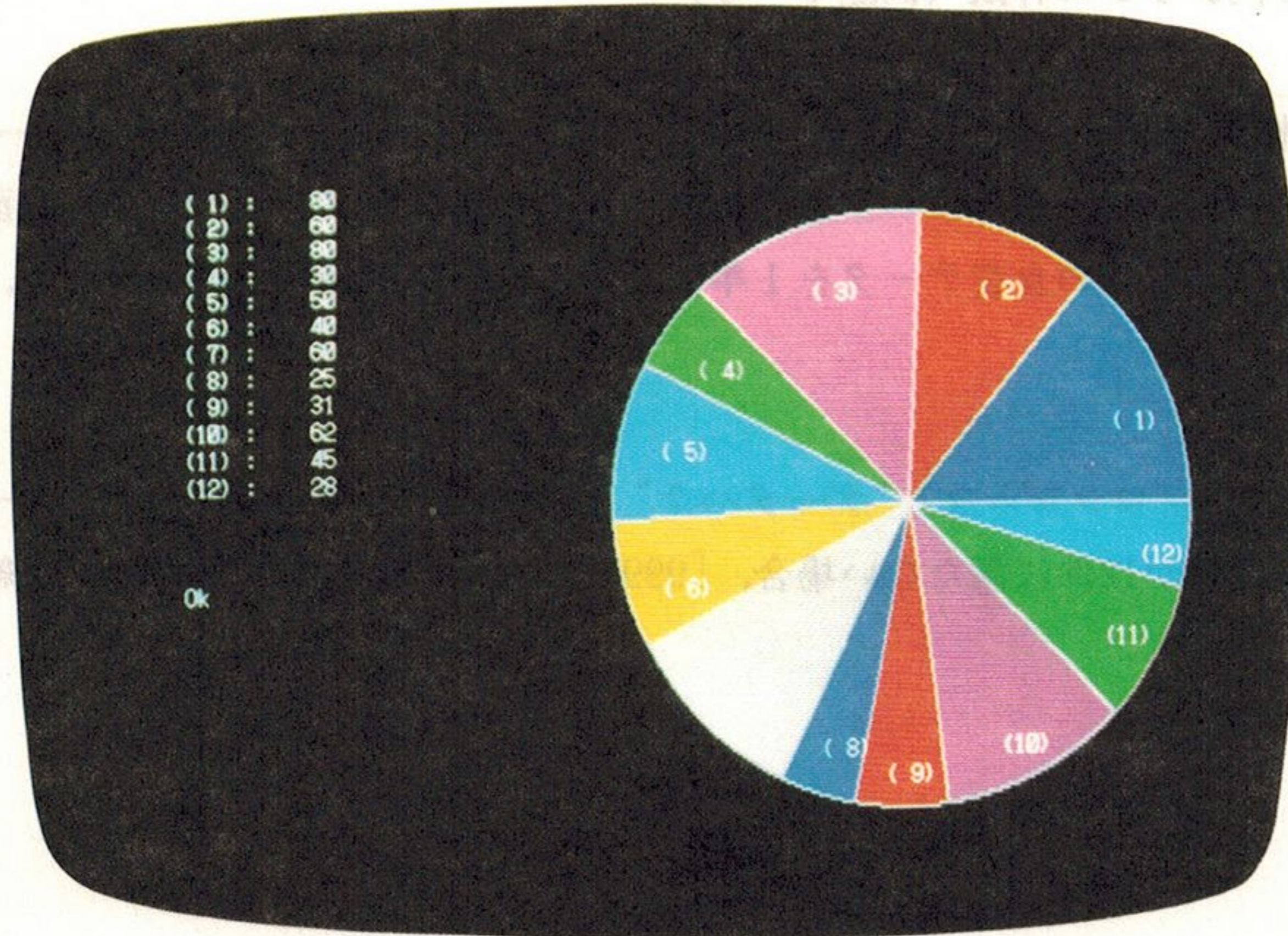
例題の概要

数値データを入力し，そのデータを分析するための棒グラフと円グラフを画面に表示します．データは，毎月の食費データを1年分，ある月の費用科目別データなど，なんでもかまいません．

◎入力方法

入力促進文（プロンプト文）“データハ？”と表示し，1件ずつデータをキー入力します．データ件数が12件に満たない場合，「99999」をキー入力するとデータが終った合図とします．

◎棒グラフ表示

左側に入力データ，右側に棒グラフを表示させます．棒グラフの目盛りは，最大値を10のべき乗（10，100，1000～）とし，それを10等分して表示させます．棒は色分けして表示させます．

◎円グラフ表示

左側に入力データ，右側に円グラフを表示させます．円グラフの各項目は，色分けし，入力データに対応させる番号をカッコ付きで表示させます．

処理の条件

入力データは，最大12件です．

・棒グラフの表示は

　① 12件のデータ入力が終ったとき

　② 「99999」が入力されたとき

に行います．

・棒グラフを数秒間表示したら，自動的に円グラフに切りかえます．

・カラーは，７色を使って青，赤，紫，緑，水色，黄色，白の順に表示させます．もし，データが８件以上ある場合，また青から順にくりかえし表示することにします．

プログラム作成上の注意点

画面にグラフを表示するには，次のことがらを知っておかねばなりません．

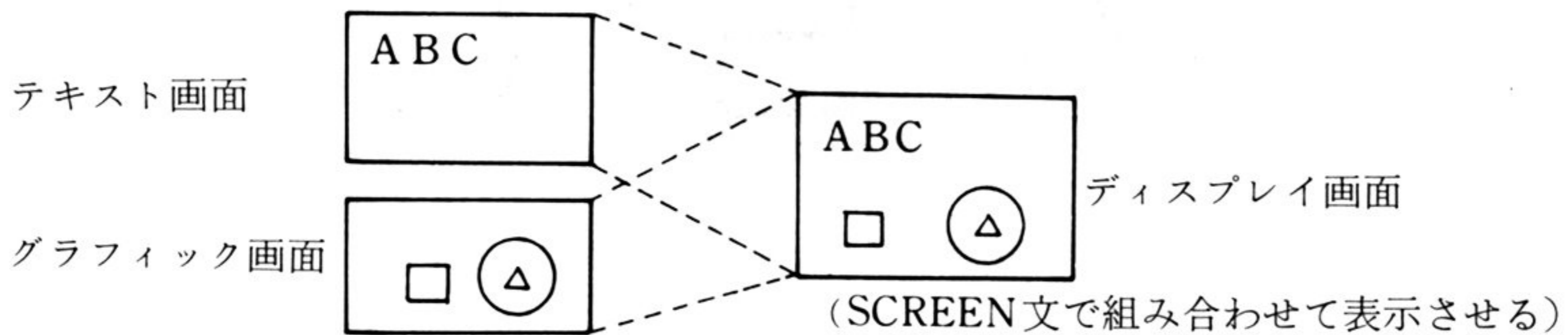
・テキスト画面，グラフィック画面の使い方

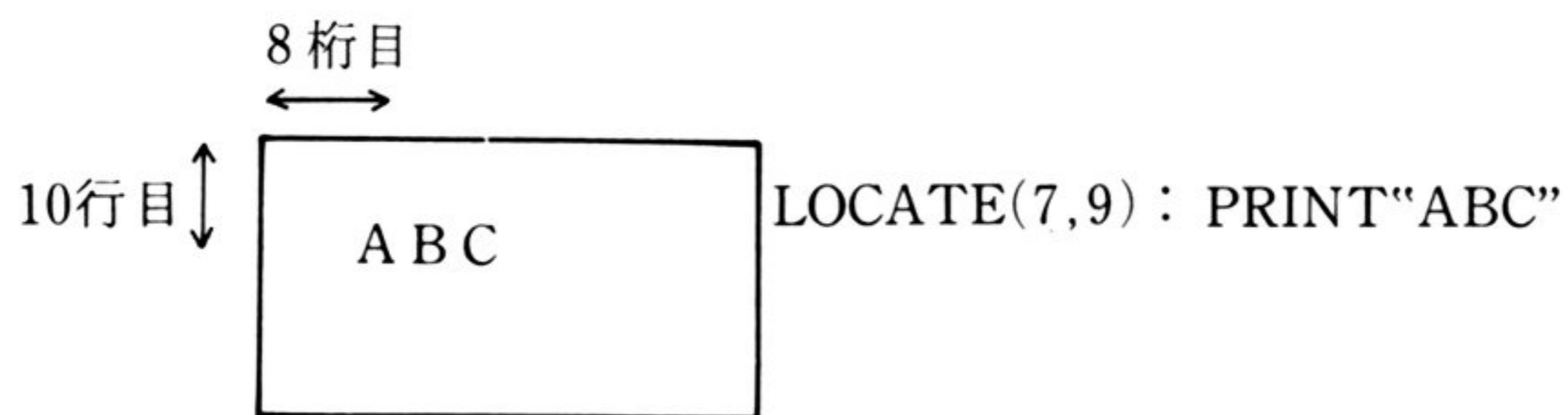
・線・四角・円の表示の仕方

・色のつけ方

ここで，その考え方を学んでおきましょう．

テキスト画面とグラフィック画面の使い方

数字や文字（グラフィックシンボルを含めて）を表示するには，テキスト画面だけあれば事足りますがテキスト画面とあわせてグラフィック画面を使用すると多様な画面作りが可能になります．

線・四角・円の表示の仕方

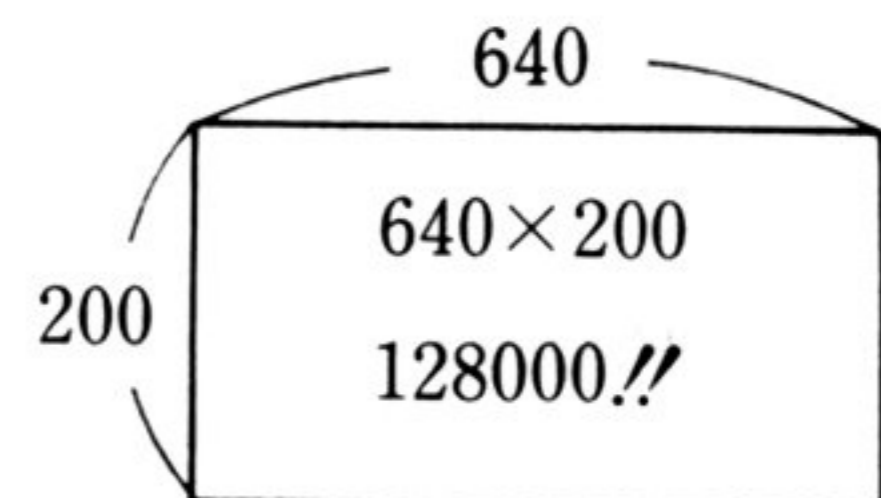
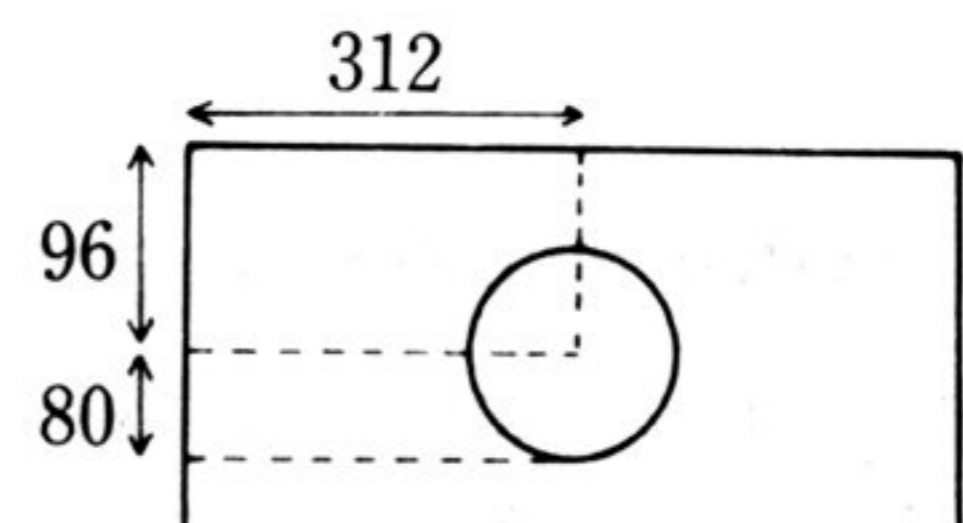
テキスト画面への文字やグラフィックシンボルの表示には，まず，LOCATE文で表示位置（左からの桁数，上からの行数）を指定し，次に，PRINT文を使って指定文字を表示させましたね．

ところが，線・四角・円などは，グラフィック画面に表示します．この場合，グラフィック画面上の表示位置つまり座標の指定が必要です．

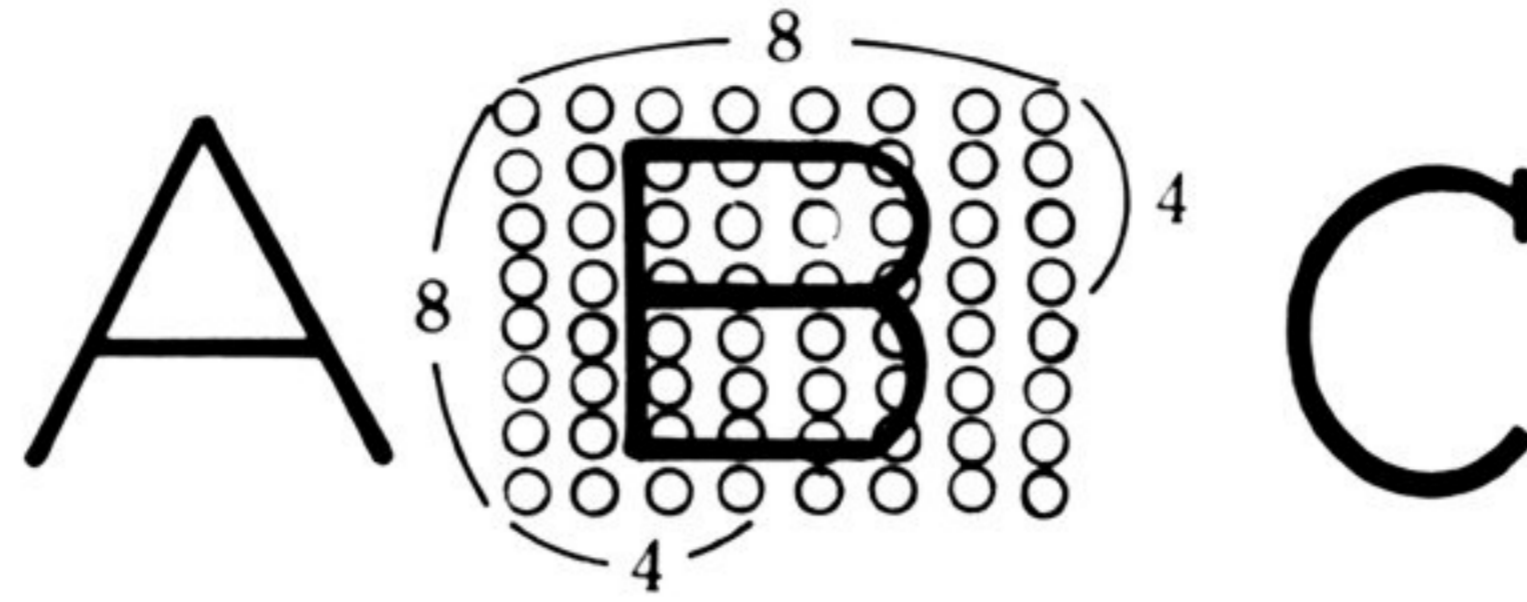
グラフィック画面では，点の表示はPSET文，線および四角の表示はLINE文，円や楕円の表示はCIRCLE文で行います．もちろん，これらの文は，すべて表示位置つまり座標が指定できる形式になっています．

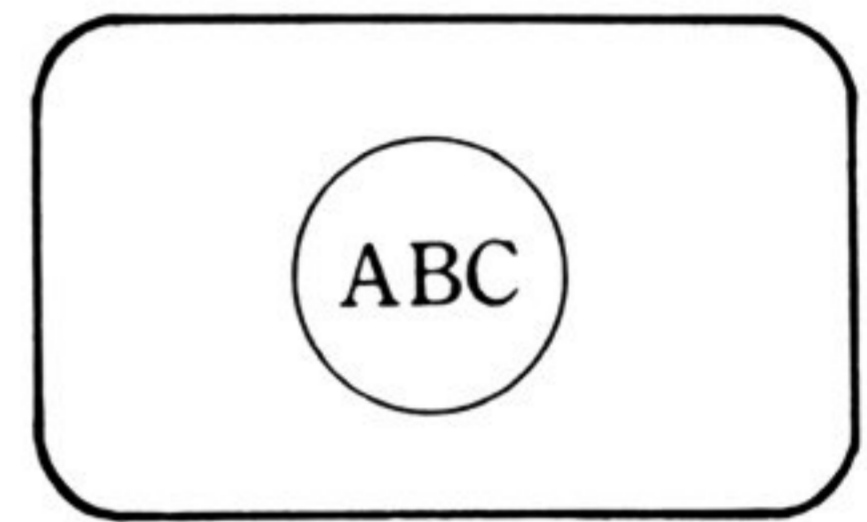
グラフィック画面の座標は，テキスト画面の座標に比べて，たて，横とも8倍の精度（たては16倍の精度のものもある）になっています．

つまり，横80桁の8倍ですから，640の点，たて25行の8倍ですから，200点のどこかということです．

640
640×200
128000!!
200

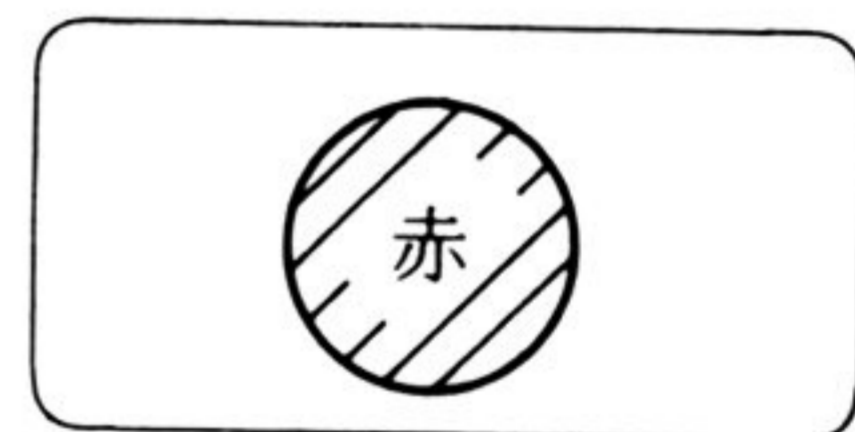
試しにやってみましょう．ディスプレイ画面の中央に半径10文字分の円を書くには，次のように指定します．

これで，横座標312，たて座標96，半径80の白の円が書けます．

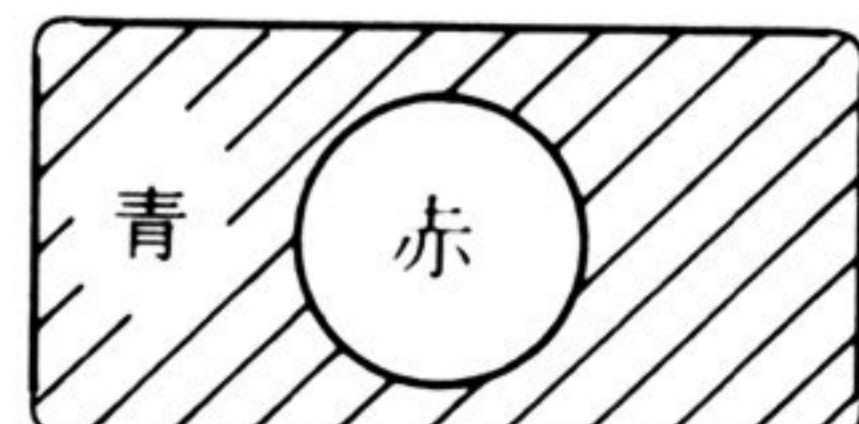
円グラフの中に文字や数字を表示するときは，テキスト画面とグラフィック画面を合成して表示することになります．グラフィック画面の座標からテキスト画面の桁数や行数を逆算するには，横座標，たて座標を８で割り算すればよいわけです．また，１文字が８つの点に対応しますから，文字の中心に合わせたいときは，グラフィック座標を４点分だけずらすことになります．

先程の円の中心に“ＡＢＣ”と表示するには，次のようにします．

LOCATE 37，12：PRINT “ABC”

12は，３文字の中心，つまり，1.5文字分（８＋４）のことで，ABCを中心にもってくるため，1.5文字分左へずらした点を指定したのです．テキスト画面は，位置指定が粗いので，割りきれないと，ちょうど中心にはなりません．

色のつけ方

とにかくやってみましょう．色を塗るにはPAINT文を使います．先程の円の中を赤く塗るには，次の様に指定します．

PAINT (312，96)，　２，　７
座標（円の内側）
２：塗る色（赤）
７：わくの色（白）

つまり，PAINT文は，指定した座標から指定した色を，わくの色のところまで塗ります．ですから，円の外側の座標（例えば（１，１））を指定してPAINT文を実行しますと，円の外側だけが塗られます．例えば，次のPAINT文では，円の外側が青くなります．

PAINT（１，１），　１，　７
座標（円の外側）
１：塗る色（青色）

PAINT文の座標を指定するとき，その座標は塗られる側に入っている点でなくてはなりません．わくの上の点を指定すると何もしてくれませんので注意してください．

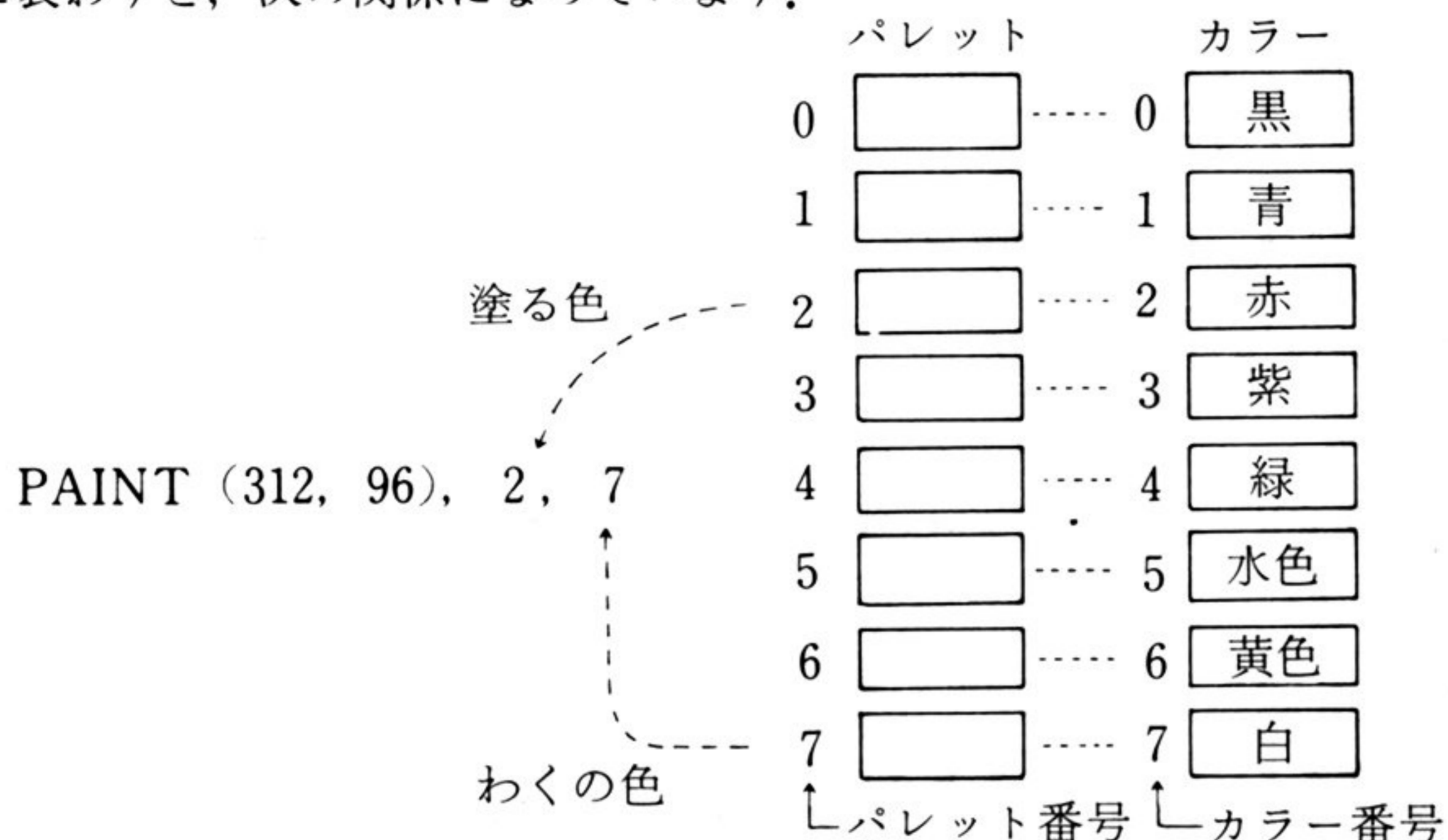
さて，塗る色の指定ですが，カラーディスプレイ画面も家庭のカラーテレビと同じように，光の三原色（青，赤，緑）を組み合わせて，いろいろな色が出せます．基本的なカラーは８色

（黒，青，赤，紫，緑，水色，黄色，白）あります．テキスト画面の文字の色は，COLOR文によって指定できます．

グラフィック画面のカラー指定は，画面モードとパレットの指定が必要です．まず，SCREEN文を使って画面モードをカラーモードに指定します．

```
SCREEN 0
       └画面モード（0のときカラー，1のとき白黒）
```

次にパレットの指定です．パレットとは，色の指定を間接的に指定させるためのものです．これは，表示する図の色を変化させるために使います．

これを図に表わすと，次の関係になっています．

カラーとパレットの対応は，その対応番号をCOLOR文（COLOR＝）で指定します．指定に当っては，１つのカラー番号をいくつのパレット番号に対応させてもかまいません．例えば，０～３のパレットは，カラー番号１（青），４～７のパレットは，カラー番号２（赤）と指定すると，塗る色を０としても，３としても青が塗られます．

パレット番号とカラー番号は，最初の状態では同じ番号に対応させてありますが，一度COLOR文で変えると，次に変えるまでその対応関係が保たれます．色指定をするプログラムでは，次の命令をあらかじめ実行しておくと，思いもよらない配色になることを防ぐことができます．

```
FOR I=0 TO 7:COLOR=(I,I):NEXT I
```

このほか，カラーは，グラフィック画面の１点ずつに指定ができますので，中間色の表現などができますが，これらについては，リファレンスマニュアルを参照して下さい．

② プログラム例

次にプログラム例を示します．実際にプログラムをキーボードより入力し，実行してみて下さい．データは，12件まで（１～32767の範囲）入力できますが，試しに10 ↵，20 ↵，80 ↵，50 ↵，70 ↵，40 ↵，99999 ↵ と入れてみて下さい．

```
10  ' ■■■■■ ｸﾞﾗﾌ ■■■■■
20  '                                                           ｶﾞﾒﾝ ｼｮｷｶ
30  CONSOLE 0,25,0
40  WIDTH 80,25
50  SCREEN 0,0,0,1
60  WINDOW (0,0)-(639,199)
70  VIEW   (0,0)-(639,199)
80  GOSUB *INPDATA
90  GOSUB *DISPDATA
100 GOSUB *BOUGRAPH
110                   FOR T = 1 TO 2000 : NEXT T
120 GOSUB *DISPDATA
130 GOSUB *ENGRAPH
140 END
150 '
160 *INPDATA          ' ■■■■■ ﾃﾞｰﾀ ﾆｭｳﾘｮｸ ■■■■■
170 DIM INDATA (12)                           ' ﾆｭｳﾘｮｸ ﾃﾞｰﾀ
180 FOR I = 1 TO 12
190   INPUT "ﾃﾞｰﾀ ﾊ"; INDATA (I)
200   IF INDATA (I) = 99999! GOTO 250
210   KENSU = KENSU + 1                       ' ﾃﾞｰﾀ ﾆｭｳﾘｮｸ ｹﾝｽｳ
220   IF INDATA(I) > S THEN S = INDATA(I)     ' ﾃﾞｰﾀ ﾉ ｻｲﾀﾞｲﾁ
230   GOKEI = GOKEI + INDATA(I)               ' ﾃﾞｰﾀ ﾉ ｺﾞｳｹｲ
240 NEXT I
250 RETURN
260 *DISPDATA         ' ■■■■■ ｶﾞﾒﾝ ｸﾘｱ, ﾃﾞｰﾀ ﾋｮｳｼﾞ ■■■■■
270 CLS      3
280 FOR I = 1 TO KENSU
290   LOCATE 0,I-1
300   PRINT  USING "(##) : #####  "; I ; INDATA(I)
310 NEXT I
320 RETURN
330 *BOUGRAPH         ' ■■■■■ ﾎﾞｰ ｸﾞﾗﾌ ■■■■■
340 BX = 184          ' ｹﾞﾝﾃﾝ ﾉ X ｻﾞﾋｮｳ
350 BY = 164          ' ｹﾞﾝﾃﾝ ﾉ Y ｻﾞﾋｮｳ
360 HABA = 22         ' ﾎﾞｳ ﾉ ﾊﾊﾞ
370 YOKO = 32         ' ﾖｺ ﾉ ｷｻﾞﾐ
380 TATE = 16         ' ﾀﾃ ﾉ ｷｻﾞﾐ
390 IF S/10 > 1 THEN C = C +  1 : S = S/10 : GOTO 390
400 MAX = 10 ^ (C + 1)     ' ﾀﾃ ｼﾞｸ ﾒﾓﾘ ﾉ ｻｲﾀﾞｲﾁ
410 '                                                      ﾀﾃ ﾖｺ ｼﾞｸ ﾋｮｳｼﾞ
420 LINE (BX,BY-TATE*10)-(BX,                BY),7
430 LINE                 -(BX+YOKO*(KENSU+1),BY),7
440 '                                                      ﾀﾃ ﾒﾓﾘ ﾉ ﾋｮｳｼﾞ
450 FOR  I = 1 TO 10
460   LOCATE BX/8-6,(BY-4)/8 - TATE/8*I
470   PRINT USING "#####";  MAX * I/10
480   LINE (BX-4,BY-TATE*I)-(BX+4,BY-TATE*I),7
490 NEXT I
500 '                                                      ﾖｺ ﾒﾓﾘ ﾉ ﾋｮｳｼﾞ
510 FOR I = 1 TO KENSU
520   LOCATE I*YOKO/8 + BX/8, (BY+4)/8
530   PRINT USING "##";I;
540 NEXT I
550 '                                                      ﾎﾞｰ ﾉ ﾋｮｳｼﾞ
560 FOR I = 1 TO KENSU
570   IF INDATA(I) = 0  THEN GOTO 620
580   TAKASA = TATE*10*INDATA(I)/MAX     'ﾀﾃｼﾞｸ ﾉ ﾀｶｻ
590   LINE (BX+I*YOKO,      BY        )
           -(BX+I*YOKO+HABA, BY-TAKASA),7,B
600   IF TAKASA < 2 GOTO 620
610   PAINT (BX+I*YOKO+1, BY-1), ((I-1) MOD 7)+1,7
620 NEXT I
630 RETURN
640 *ENGRAPH          ' ■■■■■ ｴﾝ ｸﾞﾗﾌ ■■■■■
650 EX = 440          '(EX,EY):ｴﾝ ﾉ ﾁｭｳｼﾝ ｻﾞﾋｮｳ
```

```
660 EY = 100
670 R = 180            ' ﾊﾝｹｲ
680 ST = .00001        ' ｶｲｼｶｸﾄﾞ ｦ ﾌ ﾆｽﾙﾀﾒﾉ ｺﾞｻ
690 FOR I = 1 TO KENSU
700   EN = ST + INDATA(I) / GOKEI * 3.1415 * 2
710   CIRCLE (EX,EY),R,7,-ST,-EN
720   P = EX + R/1.2 * COS(ST+(EN-ST)/2)
730   Q = EY - R/1.2 * SIN(ST+(EN-ST)/2) /2
740   PAINT (P,Q),((I-1) MOD 7)+1,7
750   LOCATE P/8-1,Q/8
760   PRINT USING "(##)";I
770   ST = EN
780 NEXT I
790 RETURN
```

プログラムを分けて，見ていきましょう．

```
10   ' ■■■■■ ｸﾞﾗﾌ ■■■■■
20   '                                        ｶﾞﾒﾝ ｼｮｷｶ
30   CONSOLE 0,25,0
40   WIDTH 80,25
50   SCREEN 0,0,0,1
60   WINDOW (0,0)-(639,199)
70   VIEW   (0,0)-(639,199)
```

ここでは，画面の初期化を行っています．グラフィック画面を使うときは，必ず初期化しなければなりません．各文の詳細は後述しますが，意味の概略を説明しておきましょう．

「CONSOLE（コンソール）」文で，テキスト画面のスクロールする領域を0行から25行（画面全体）とするとともに，画面の最下段のファンクションキーの表示を消しています．スクロールする領域とは文字が表示できる範囲です．今回は，画面全体に表示します．ファンクションキーを表示したままでもグラフ表示はできますが，消した方が見栄えがします．（再度表示したいときは，CONSOLE,,1とします）

「WIDTH（ウィズ）」文で，テキスト画面に表示する文字の桁数，行数を指定しています．この場合は，最大の80桁と25行を指定しました．

「SCREEN（スクリーン）」文は，グラフィック画面をカラーモードにするとともに，普通の速度で表示する指定を行っています．

「WINDOW（ウインドウ）」関数と「VIEW（ビュー）」関数で，グラフィック画面の座標の範囲と，ディスプレイ画面の座標の範囲を指定しています．両画面とも座標は0から始まりますから，横(X)方向に640点，たて(Y)方向に200点を座標範囲として指定してあります．通常このように指定しますが，詳しくは，リファレンスマニュアルを参照して下さい．

```
80  GOSUB *INPDATA
90  GOSUB *DISPDATA
100 GOSUB *BOUGRAPH
110               FOR T = 1 TO 2000 : NEXT T
120 GOSUB *DISPDATA
130 GOSUB *ENGRAPH
140 END
150 '
```

このプログラムは，第6章で学んだサブルーチンを使って，プログラムを4つに分けています．各サブルーチンは，次のことを実行します．

＊INPDATA ……データを最大12件まで読み，配列に記憶する，かつデータの最大値，および全データの合計を求める．

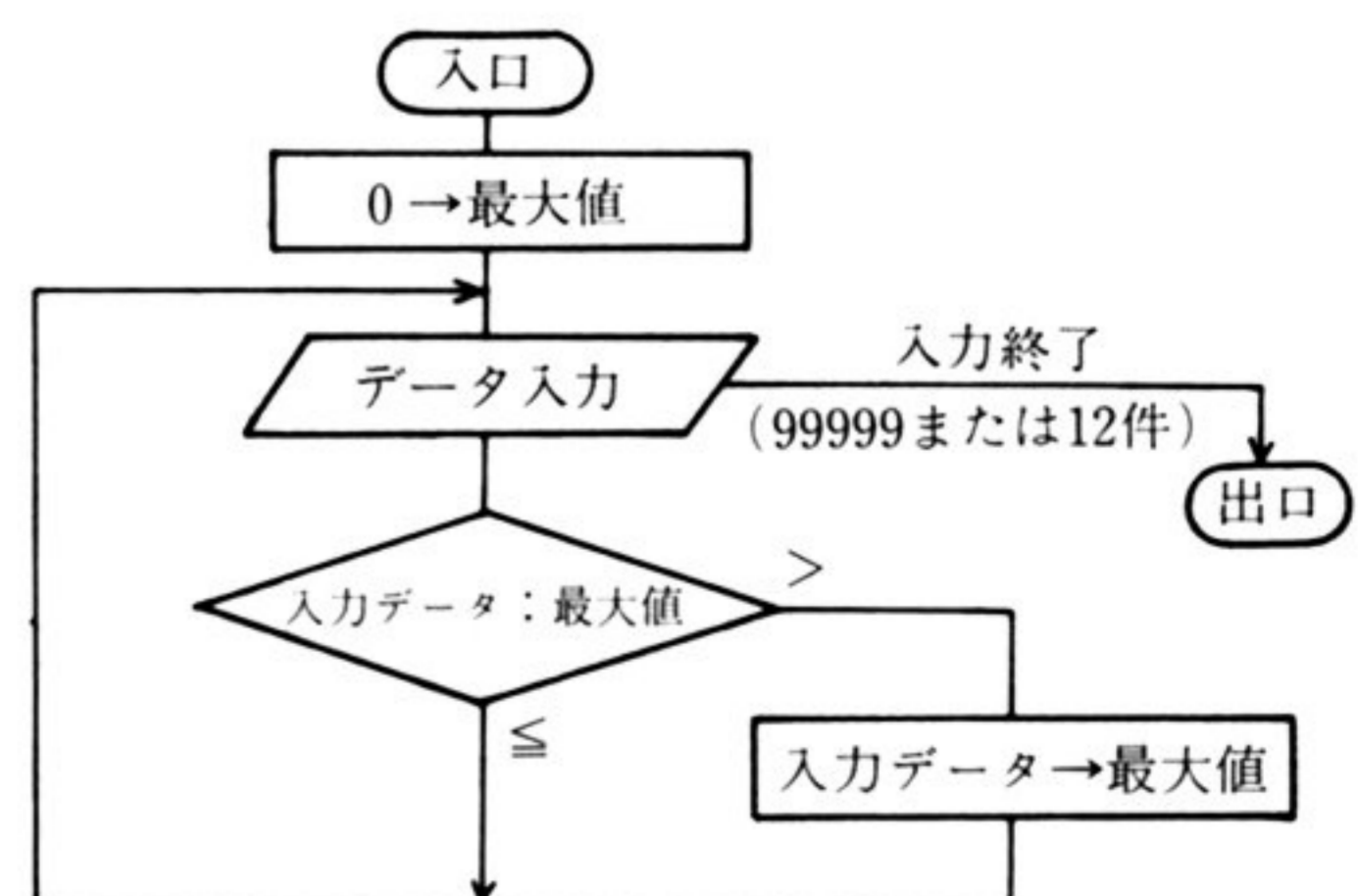
＊DISPDATA……入力されたデータを画面に表示する．

＊BOUGRAPH……棒グラフを表示する．

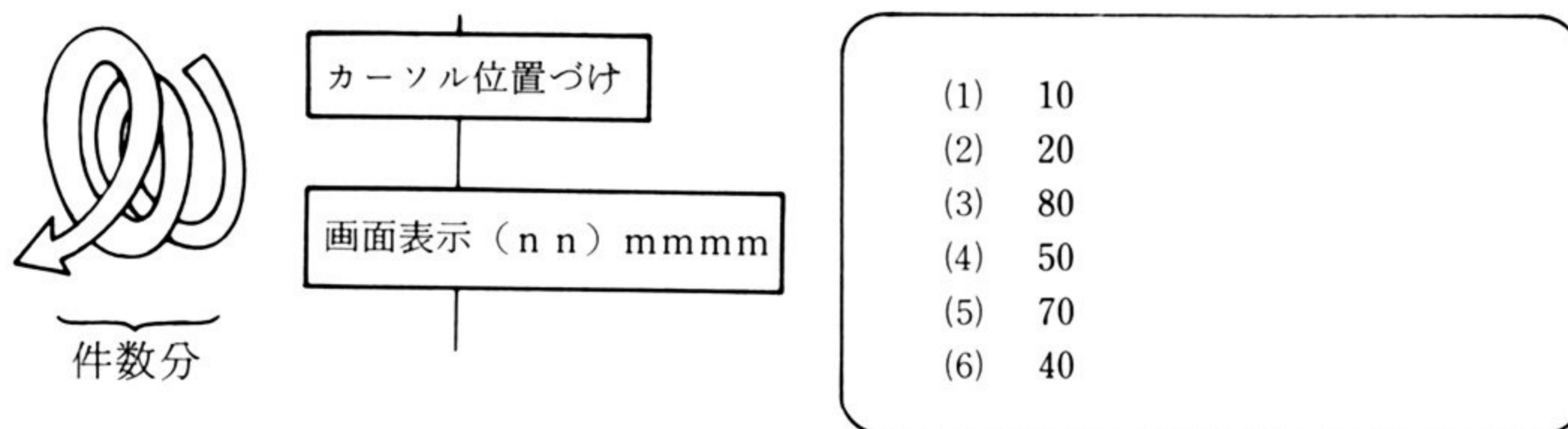
＊ENGRAPH ……円グラフを表示する．

真中の「FOR」文は，棒グラフを一定の時間表示させておくためのものです．この場合は，本体の演算速度で，数を1から始めて2000まで数えています．つまり，表示時間は，数値を大きくすれば長く，小さくすれば短くなります．「FOR」文のかわりにINPUT A$とすると，何らかのデータがキー入力されるまで，棒グラフは表示されています．

```
160 *INPDATA          ' ■■■■ デ゛ータ  ニュウリョク ■■■■
170 DIM INDATA (12)                              ' ニュウリョク デ゛ータ
180 FOR I = 1 TO 12
190   INPUT "デ゛ータ ハ"; INDATA (I)
200   IF INDATA (I) = 99999! GOTO 250
210   KENSU = KENSU + 1                          ' デ゛ータ ニュウリョク ケンスウ
220   IF INDATA(I) > S THEN S = INDATA(I)       ' デ゛ータ ノ サイダ゛イチ
230   GOKEI = GOKEI + INDATA(I)                 ' デ゛ータ ノ ゴ゛ウケイ
240 NEXT I
250 RETURN
```

ここでは，キー入力されたデータを12個までの配列変数として記憶します．配列変数については，第6章で詳しく説明しました．記憶したデータの件数，データの最大値，データの合計も記憶します．最大値を求めるには，まず0を最大値と仮定しデータが入力されるたびに，最大値と比較し，もし，入力されたデータの方が大きければ，最大値と置きかえるという操作をくりかえします．データ「99999」，または12件のデータが入力されると最大値という変数の中に最大のデータが残ります．

```
260 *DISPDATA          ' ━━━━━ ｶﾞﾒﾝ ｸﾘｱ, ﾃﾞｰﾀ ﾋｮｳｼﾞ ━━━━━
270 CLS       3
280 FOR I = 1 TO KENSU
290   LOCATE 0,I-1
300   PRINT  USING "(##) : #####  "; I ; INDATA(I)
310 NEXT I
320 RETURN
```

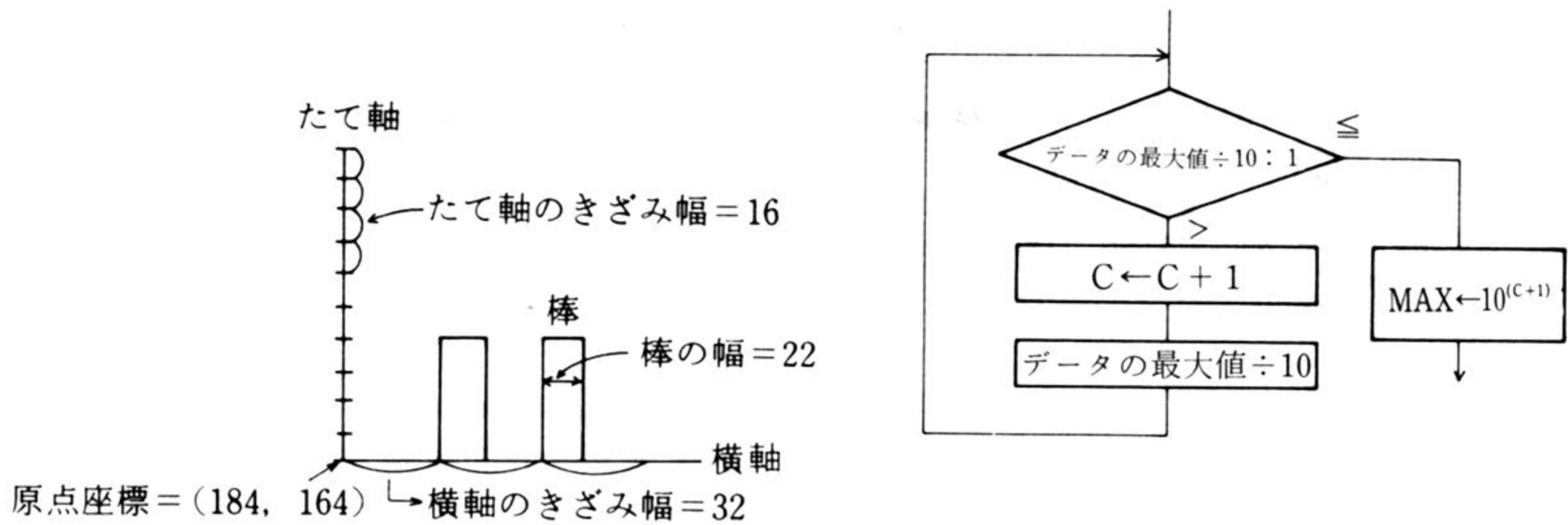
いよいよグラフを表示するわけですが，その前に画面を消さなくてはなりません．画面を消すには，「CLS（クリアスクリーン）」文を使います．「CLS」文は，そのパラメータが1ならテキスト画面，2ならばグラフィック画面，3のときは両方の画面を消します．

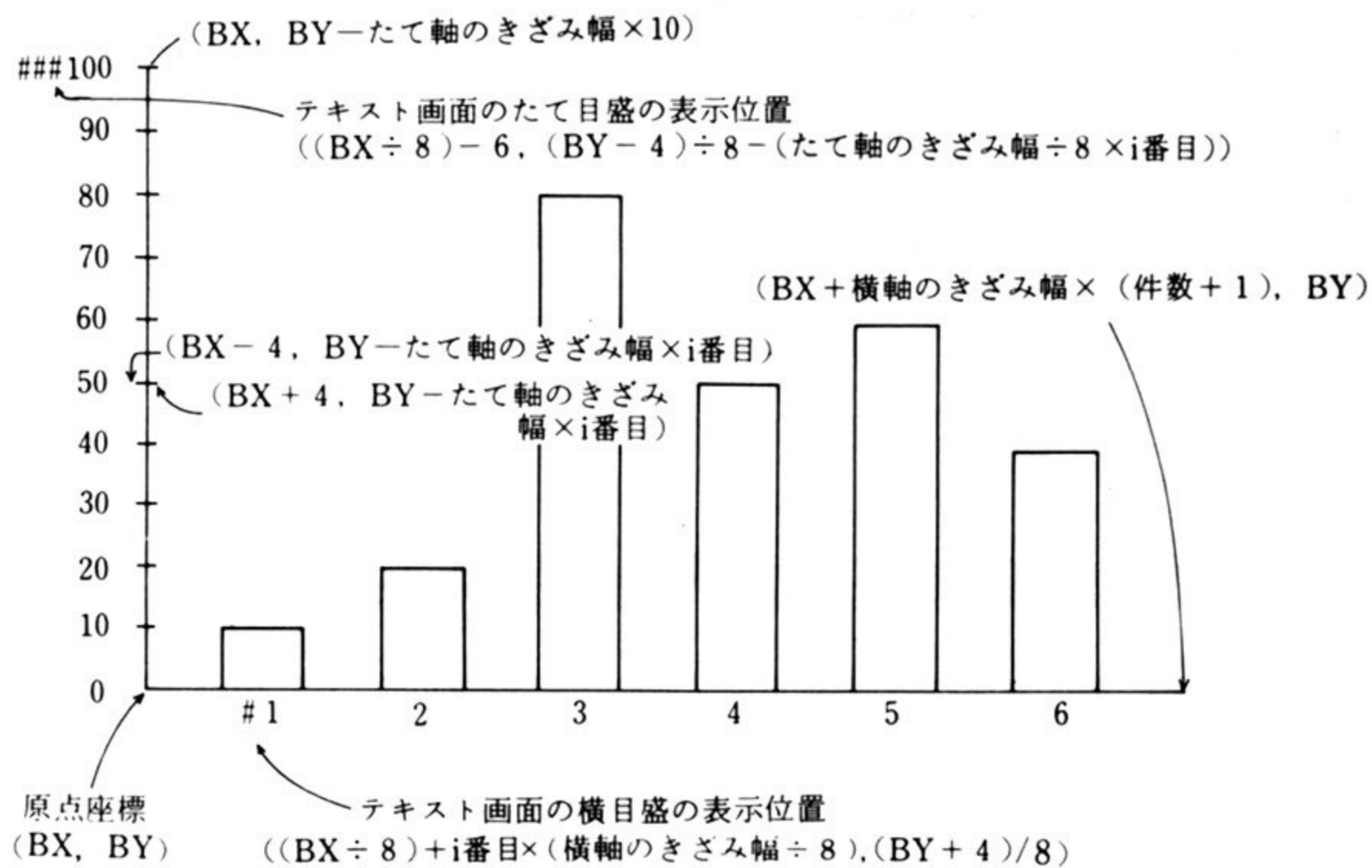
続けて，「LOCATE（ロケート）」文でカーソルを位置づけ，「PRINT」文で入力データを1件ずつ改行しながら，全件数分表示します．

```
330 *BOUGRAPH         ' ━━━━━ ﾎﾞｰ ｸﾞﾗﾌ ━━━━━
340 BX = 184          ' ｹﾞﾝﾃﾝ ﾉ X ｻﾞﾋｮｳ
350 BY = 164          ' ｹﾞﾝﾃﾝ ﾉ Y ｻﾞﾋｮｳ
360 HABA = 22         ' ﾎﾞｳ ﾉ ﾊﾊﾞ
370 YOKO = 32         ' ﾖｺ ﾉ ｷｻﾞﾐ
380 TATE = 16         ' ﾀﾃ ﾉ ｷｻﾞﾐ
390 IF S/10 > 1 THEN C = C + 1 : S = S/10 : GOTO 390
400 MAX = 10 ^ (C + 1)    ' ﾀﾃ ｼﾞｸ ﾒﾓﾘ ﾉ ｻｲﾀﾞｲﾁ
```

さて，棒グラフの表示を，次の3つに分けて説明しましょう．

- 前準備
- たて軸／横軸の表示
- 棒の表示

まず，前準備です．ここでは，棒グラフの原点座標(BX，BY)と，表示する棒の幅(HABA)，たて軸のきざみ幅（TATE)，横軸のきざみ幅（YOKO）のそれぞれについて，グラフィック画面上の点の座標，および，点の数を決めています．

また，たて軸は，10個のきざみとし，10のべき乗数の目盛になるよう最大値（MAX）の値を求めています．つまり，データが10，20，80，50，70，40のように全部2桁以下であれば，目盛の最大値を100，3桁以下であれば，目盛の最大値が1000となるようにしています．

```
410 '                                                  タテ ヨコ ジク ヒョウジ
420 LINE (BX,BY-TATE*10)-(BX,               BY),7
430 LINE                -(BX+YOKO*(KENSU+1),BY),7
440 '                                                  タテ メモリ ノ ヒョウジ
450 FOR I = 1 TO 10
460   LOCATE BX/8-6,(BY-4)/8 - TATE/8*I
470   PRINT USING "#####";  MAX * I/10
480   LINE (BX-4,BY-TATE*I)-(BX+4,BY-TATE*I),7
490 NEXT I
500 '                                                  ヨコ メモリ ノ ヒョウジ
510 FOR I = 1 TO KENSU
520   LOCATE I*YOKO/8 + BX/8, (BY+4)/8
530   PRINT USING "##";I;
540 NEXT I
```

ここでは，棒グラフのたて軸，横軸，および，たて軸の目盛り，横軸の目盛りを表示します．それぞれの座標は，次の図のようになります．

この図を見て，各点の座標を確認して下さい．各点の座標は，原点（BX，BY）からの相対位置になっています．また，数字は，テキスト画面で表示してグラフィック画面に重ねて表示します．テキスト画面上の位置は，「LOCATE（ロケート）」文で指定します．グラフィック画面とテキスト画面は，横が200対25，たてが640対80でそれぞれ8対1の関係になっているので，それぞれ8で割っています．グラフィック画面上の4点分は，テキスト画面上の文字半分ですので，目盛では4増減させています．

より便利に

画面上にグラフを表示するために座標を計算し，プログラムとして記述するとき，50とか100というように直接数値で指定することが多くあります．しかし，めんどうでも，このプログラムのように，変数を使って相対的に表現しておくと，表示位置を変更させたり，大きさをかえたり，ちょっと直して応用することが自由にできます．

原点の座標やきざみ幅，棒の幅など変更して実行してみましょう．

```
550 '                                                    ﾎﾞｰﾉ ﾋｮｳｼﾞ
560 FOR I = 1 TO KENSU
570   IF INDATA(I) = 0  THEN GOTO 620
580   TAKASA = TATE*10*INDATA(I)/MAX      'ﾀﾃｼﾞｸ ﾉ ﾀｶｻ
590   LINE (BX+I*YOKO,        BY        )
         -(BX+I*YOKO+HABA, BY-TAKASA),7,B
600   IF TAKASA < 2 GOTO 620
610   PAINT (BX+I*YOKO+1, BY-1), ((I-1) MOD 7)+1,7
620 NEXT I
630 RETURN
```

さて，いよいよ棒の部分を表示します．棒は，２点を指定しその２点を頂点とする四角として表示させます．「LINE」文は，本来，２点を結ぶ線をグラフィック画面上に表示する命令ですが，最後のパラメータに「B」を指定すると指定された２点を相対する頂点とする四角形を表示します．

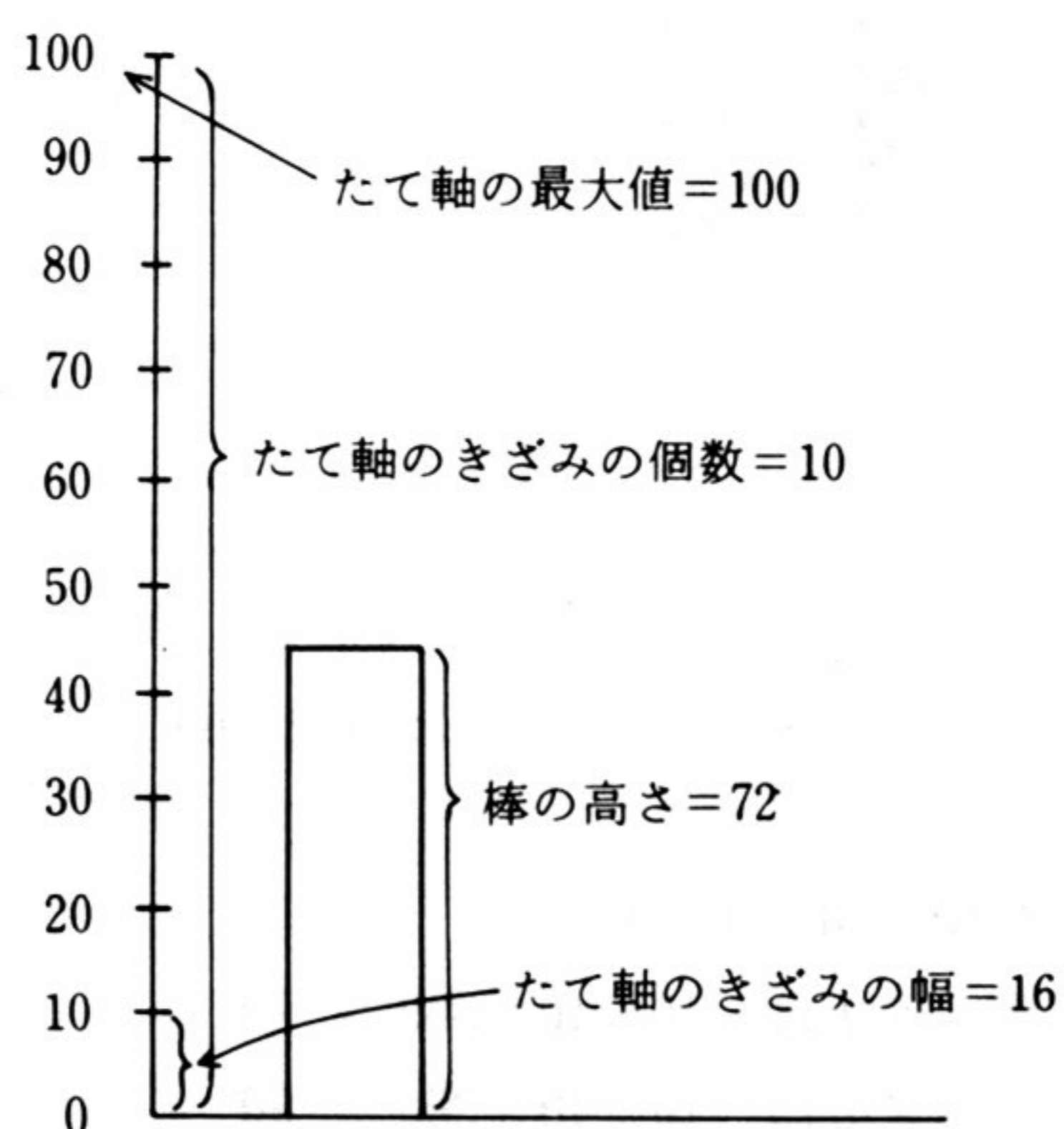
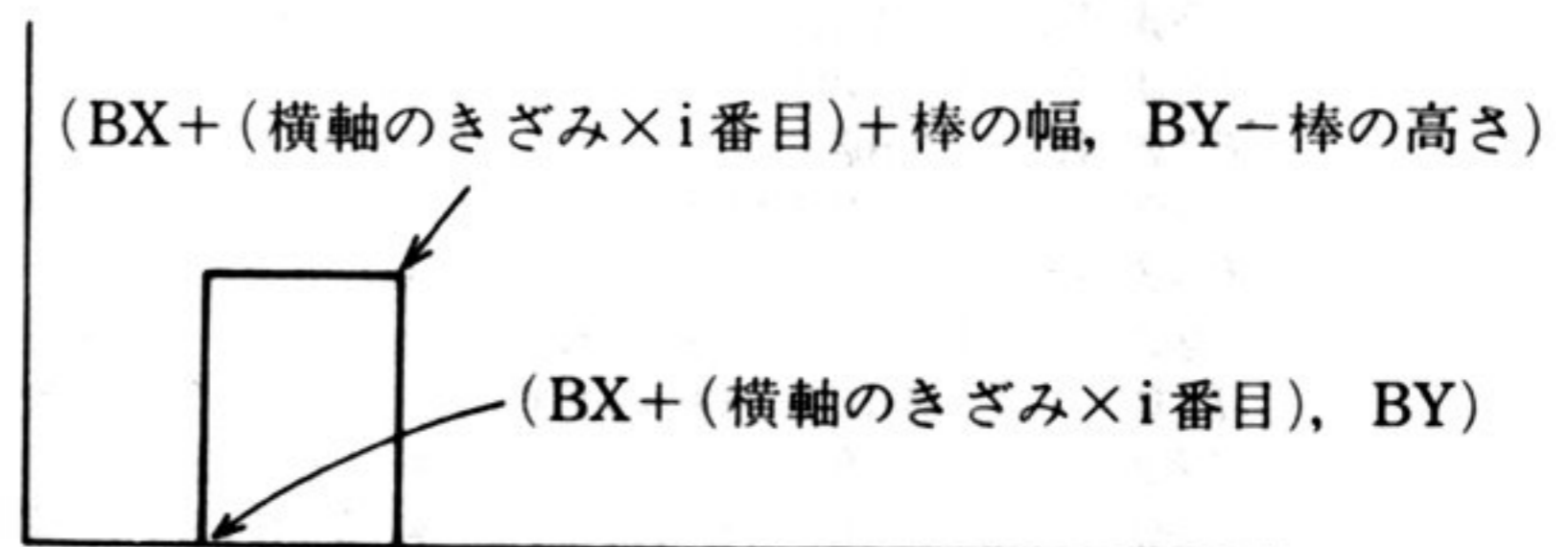
まず，データが０かチェックし，０なら棒の部分の表示は必要ないので，次のデータに進みます．

次に，棒の座標ですが，図のようになります．

ここで棒の高さは，たて軸の最大値に，各データの高さの割り合いをかけ，それに，たて軸のきざみ幅ときざみの個数をかけたものです．たとえば，たて軸の最大値が100，たて軸のきざみが16，きざみの個数が10，データが45とすると，「たて軸の１メモリ当りの高さの割り合い」は，

最大値÷たて軸のきざみ個数

100 ÷ 10 ＝ 10

であり，「棒の高さ」は

たて軸のきざみ幅×データ÷（たて軸の１メモリ当りの高さの割り合い）

$$16 \times 45 \div 10 = 72$$

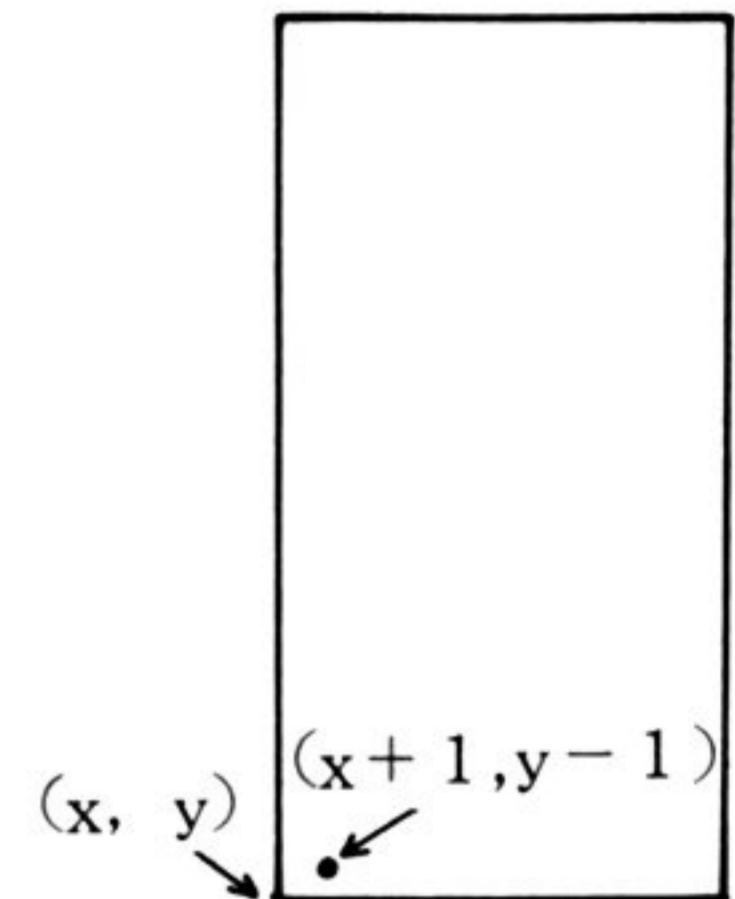
となります．なお，このプログラムでは，たて軸のきざみの個数は10個と固定です．棒の部分が表示できたら，色をつけます．色は「PAINT」文でつけることができます．「PAINT」文は，ある色のわくで囲まれた部分を，指定された色で塗ります．「PAINT」文は，次の様に指定します．

PAINT（X座標，Y座標），塗る色のパレット番号，わくの色のパレット番号

座標は，わくの内側の点を指定します．そこで，まず棒の高さがある（２以上）かチェックし，２以上なら色を塗ります．わくの内側の点ですから，横軸方向に＋１，たて軸方向に－１しています．

次に塗る色のパレット番号ですが，１から入力件数分までかわる間，Ｉが１から７の数をくりかえすように指定します．

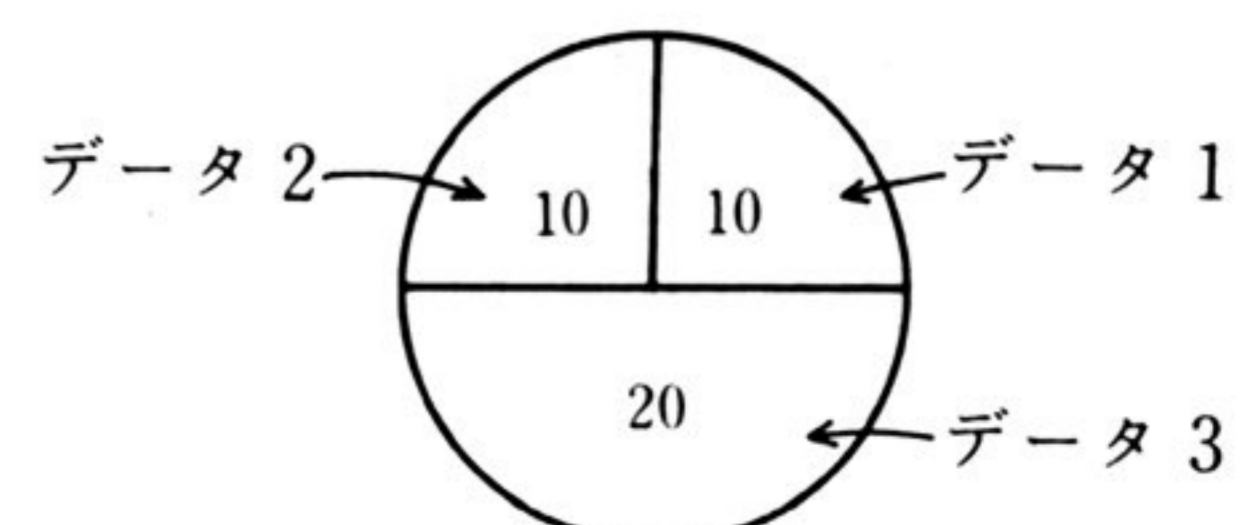
MODは，除算の余りを求める演算子です．

たとえば，9 MOD 7は，９÷７＝１…２ですから，答は２になります．同様に0 MOD 7は，０÷７＝０…０で答は０です．よって（(Ｉ－１) MOD 7)＋１は，Ｉが１，２，３，…，７，８，９のとき，１，２，３，…，７，１，２，となります．

```
640 *ENGRAPH          ' ■■■■■ エン グラフ ■■■■■
650 EX = 440          '(EX,EY);エン ノ チュウシン ザヒョウ
660 EY = 100
670 R = 180           ' ハンケイ
680 ST = .00001       ' カイシカクド ヲ フ ニスルタメノ ゴサ
690 FOR I = 1 TO KENSU
700   EN = ST + INDATA(I) / GOKEI * 3.1415 * 2
710   CIRCLE (EX,EY),R,7,-ST,-EN
720   P = EX + R/1.2 * COS(ST+(EN-ST)/2)
730   Q = EY + R/1.2 * SIN(ST+(EN-ST)/2) /2
740   PAINT (P,Q),((I-1) MOD 7)+1,7
750   LOCATE P/8-1,Q/8
760   PRINT USING "(##)";I
770   ST = EN
780 NEXT I
790 RETURN
```

さて，円グラフに移りましょう．まず，円グラフというものについて考えてみましょう．データが３件，10，10，20という例で円グラフを描くと図のようになります．円グラフは，1周(360°) をデータの比率で割って表わします．

つまり10＋10＋20＝40

データ１は　10÷40×360°＝ 90°

データ２は　10÷40×360°＝ 90°

データ3は　20÷40×360°＝180°

です．円は，その中心座標と半径を決めれば，「CIRCLE」文を使って簡単に表示することができます．例えば，円の中心座標が（200，100）で半径が50の円は，次の指定になります．実際に試してみて下さい．

CIRCLE（200，1000），50，7

一方，円弧を表示する場合，これに開始角度と終了角度をラジアンという単位で指定します．

ラジアンという単位は，図の様になっています．π（パイ）は，3.1415です．右上の弧を表示するには，次の指定になります．

（ラジアン）

開始角度が0，終了角度が1/2πつまり3.1415/2ですから，

CIRCLE（200，100），50，7，0，3.1415/2

扇形の場合，開始角度と終了角度にマイナスをつけます．0には正も負もありません．そこで0が開始角度の場合，十分に小さい数（0.0001）を指定してマイナスをつけます．

（弧）

CIRCLE（200，100），50，7，−0.0001，−3.1415／2

円グラフは，2πを100として，各データの割合いを扇形に表示してゆけば作成できます．各扇形の終了角度は，次のようになります．

$$終了角度＝開始角度＋\frac{データ}{データの合計}×2\pi$$

（扇形）

続いて，扇形に色を塗るために扇形の中の点を三角関数（SIN，COS）を使って計算しましょう．

三角関数の公式は，

A＝C×（COSθ）

B＝C×（SINθ）

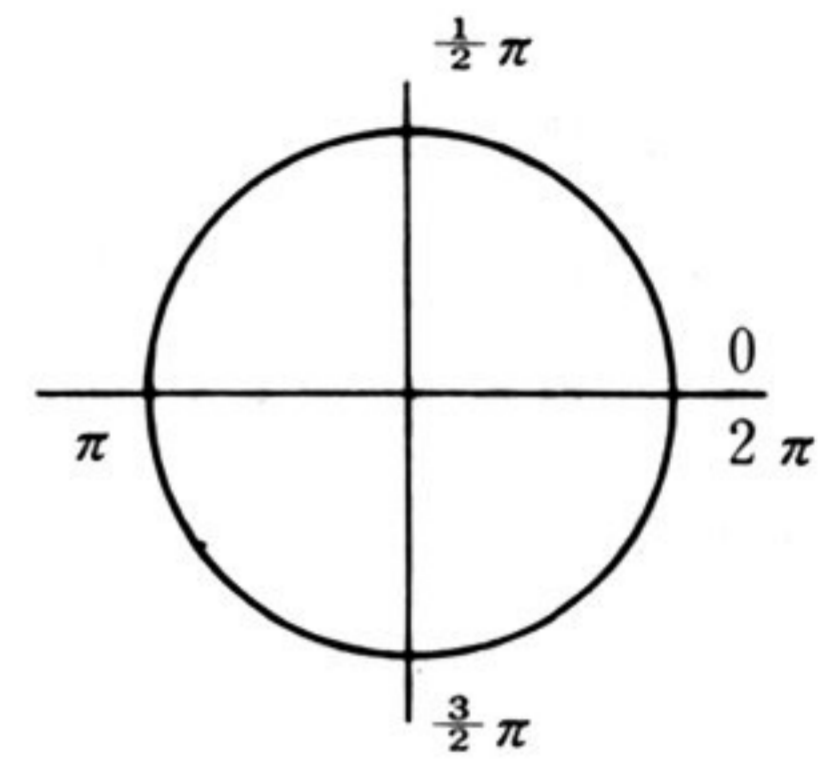
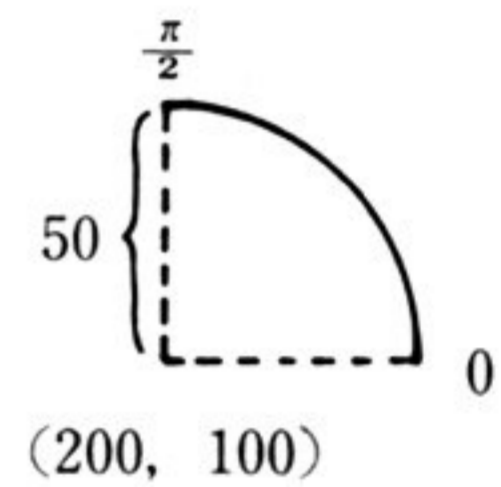
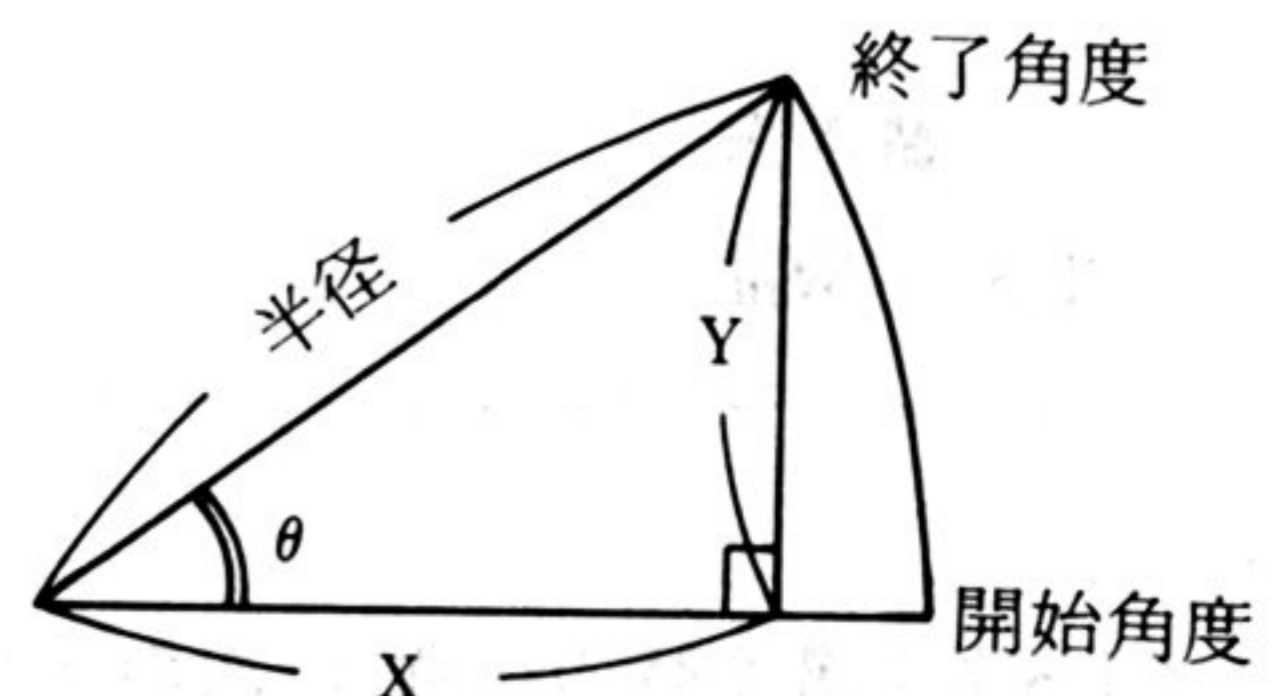
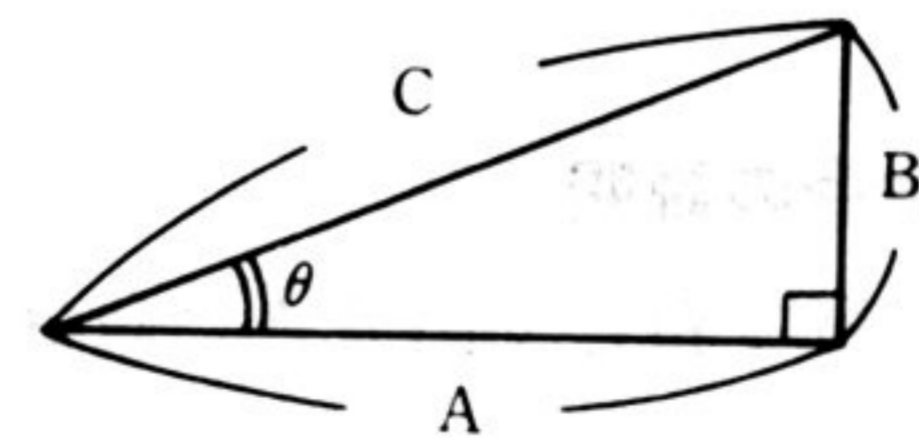
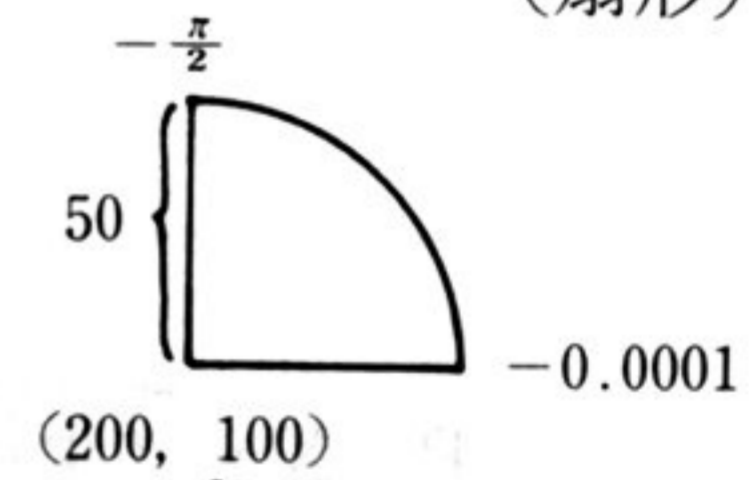
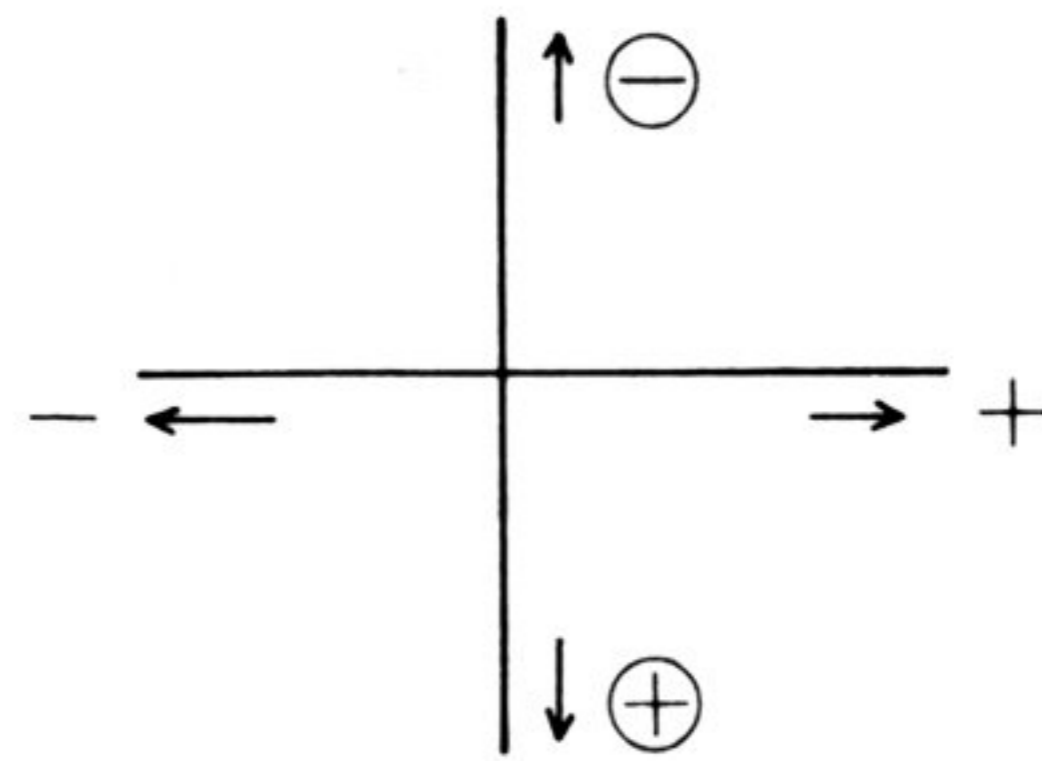
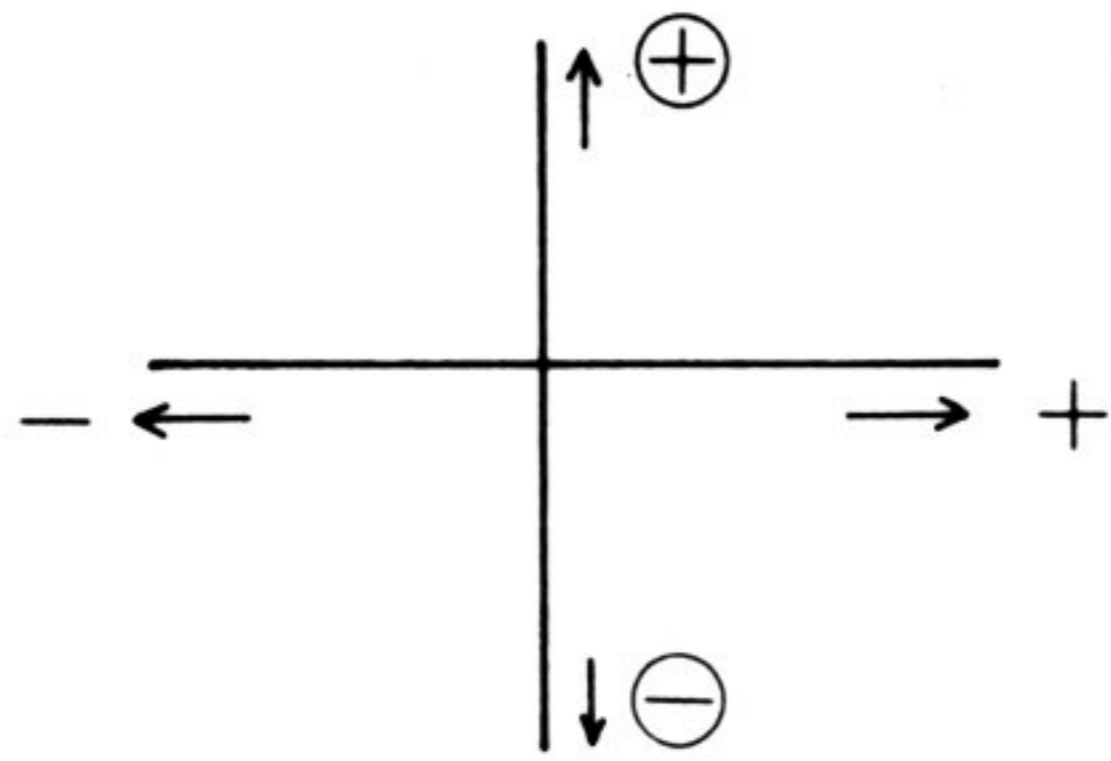
です．つまり，扇形の弧の終りの点は，次のようになります．

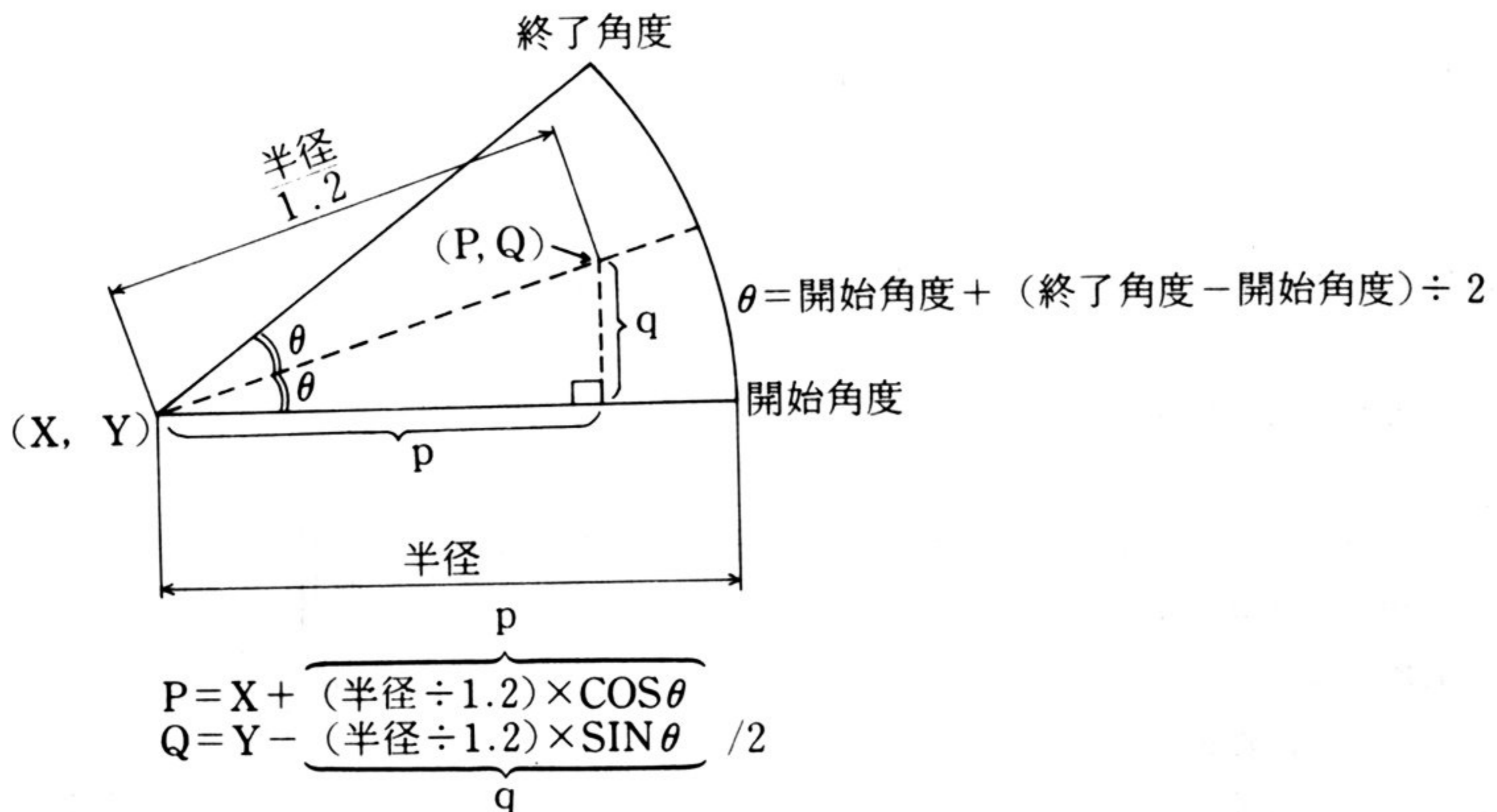
X座標：半径×COSθ

Y座標：−（半径×SINθ）

（θ：終了角度−開始角度）

ここで，X，Y座標を計算するためには，グラフィック画面の特徴をよく理解しておく必要があります．1つは，Y座標が数学の公式などとは正負が反対になっていることです．

グラフィック画面上の座標

もう1つは，X軸とY軸の座標は，ディスプレイ画面の分解能が（0，0）−（639，399）のとき1対1に対応しています．ここでは，（0，0）−（639，199）の範囲を使いますから，Y軸の座標計算に当っては，X軸の半分となります．

「PAINT」文の座標は扇形の中のどこでもよいのですが，その中に表示する対応番号がうまく表示できる点として，半径の1.2分の一の点にしました．プログラムでは，扇形を表示した後，COS関数，SIN関数を使って，扇形の中の座標を求め，「PAINT」文を出すとともに，「LOCATE」文で，その座標に当るテキスト画面の表示位置にカーソルを位置づけ，「PAINT」文で対応番号を表示しています．

3 命令の説明

この章で新しく出てきた命令を，見ていくことにしましょう．

① CLS文

テキスト画面，グラフィック画面をクリアします．

書き方

CLS〔〈機能〉〕

CLS文を書く時の注意は，次のとおりです．

・〈機能〉は，1，2，3の数字で指定します．
　1は，テキスト画面をクリアします．
　2は，グラフィック画面をクリアします．
　3は，テキスト画面とグラフィック画面をクリアします．

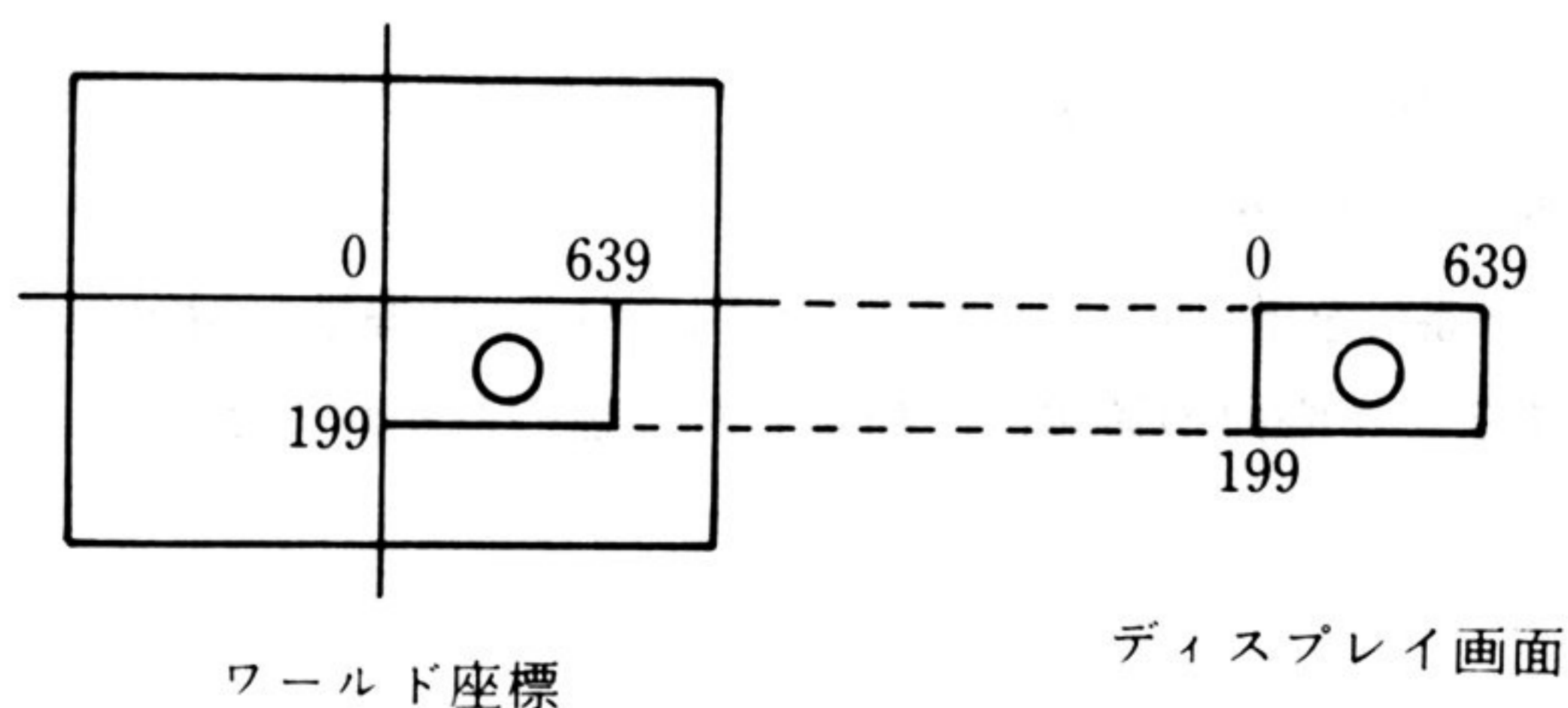
② WINDOW文

ディスプレイ画面に表示するワールド座標系の領域を指定します．

書き方

WINDOW文（X_1，Y_1）－（X_2，Y_2）

WINDOW文を書く時の注意は次のとおりです．

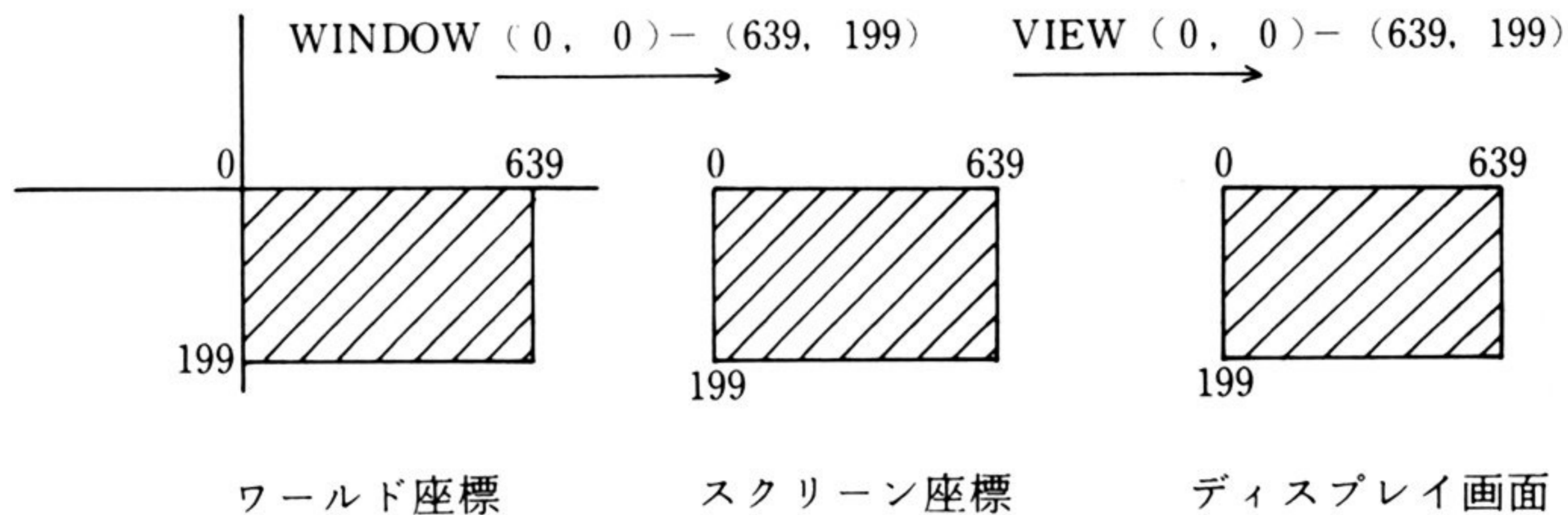
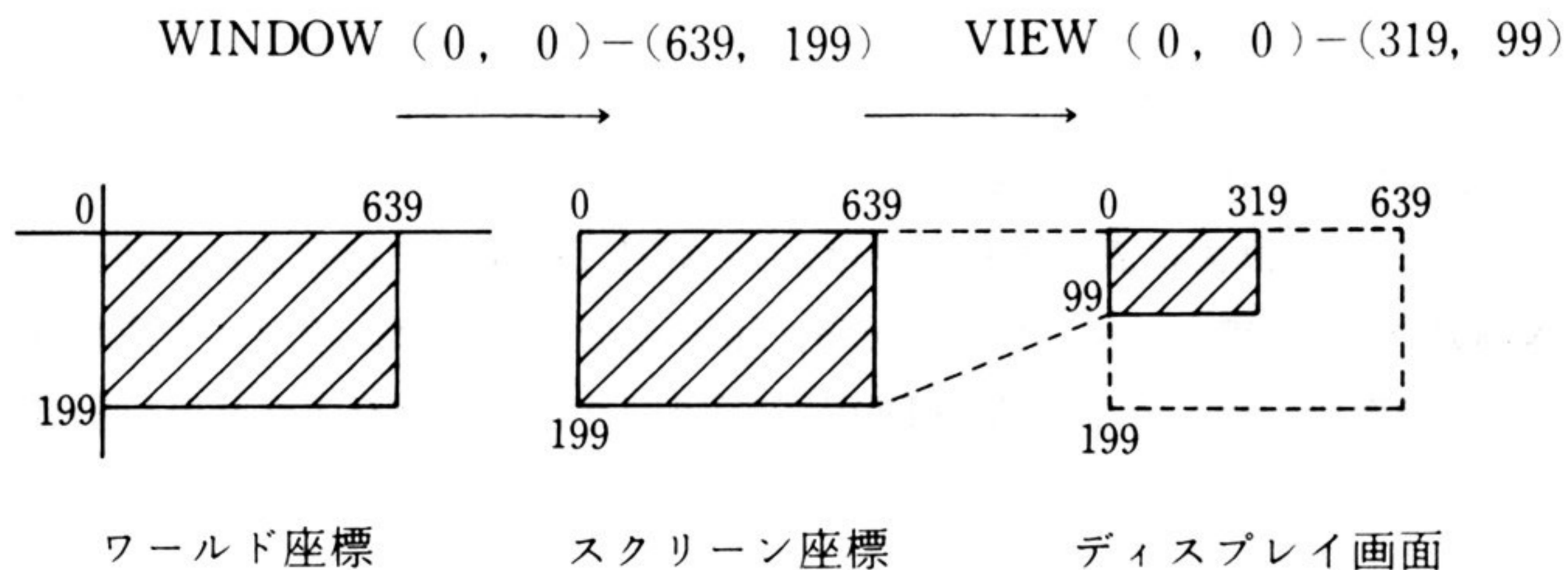
N_{88}－BASIC(86)では，図形(グラフィックス)を扱うために大きな仮想座標系を持っており，それをワールド座標系と呼びます．図形は，このワールド座標系の中で扱うことになっていますが，ディスプレイ画面に表示されるのは，このWINDOW文で指定された範囲のものです．WINDOW文で座標を（0，0）－（639，199）と指定すれば，ワールド座標とディスプレイ画面の座標が一致します．WINDOW文で限定された座標をスクリーン座標と呼びます．

③ VIEW文

ディスプレイ画面のうち，グラフィック画面を表示する領域を指定します．

書き方

VIEW（X_1，Y_1）－（X_2，Y_2）〔，〈領域色〉〕〔，〈境界色〉〕

VIEW文を書く時の注意は，次のとおりです．

・WINDOW文で指定した領域をディスプレイ画面のどの部分に表示するかを指定します．VIEW（0，0）－（639，199）と指定すると，ディスプレイ画面全体に表示します．

・VIEW（0，0）－（319，99）と指定するとグラフィック画面は，ディスプレイ画面の左上四半分に縮少されて表示されます．

・領域色はパレット番号を指定します．パレット番号が指定されると，VIEW文で指定したディスプレイ画面の領域の地色（クリアされたときの色）が，その指定された色になります．

・境界色はパレット番号を指定することができます．パレット番号が指定されると，VIEW文で指定したディスプレイ画面の領域に，指定された色のわくがつきます．

④　SCREEN文

グラフィック画面の種々のモードを指定します．

書き方

SCREEN〔<画面モード>〕〔，<画面スイッチ>〕
　　〔，<アクティブページ>〔，<ディスプレイページ>〕

SCREEN文を書く時の注意は，次のとおりです．

・画面モードは，グラフィック画面のカラーやディスプレイの分解能の指定をします．通常，0を指定します．

画面モード	：0	カラーモード
〃	：1	白黒モード
〃	：2	高分解能白黒モード
〃	：3	高分解能カラーモード

・画面スイッチは，グラフィック画面を一時的に消したり，表示速度を高速にしたりするための指定をします．通常，0を指定します．

画面スイッチ	：0	グラフィック画面を通常速度で表示	
〃	：1	〃	を速い速度で表示
〃	：2	〃	を一時的に消す
〃	：3	〃	を一時的に消すとともに，高速で入れかえる．

・アクティブページ，ディスプレイページの指定は，カラーモードでは通常0，1を指定します．詳しくは，リファレンスマニュアルを参照して下さい．

⑤ LINE文

指定された2点間に直線を引きます．

書き方

LINE 〔｛(X1，Y1) ｜ STEP (X1，Y1)｝〕 ｛(X2，Y2) ｜ STEP (X2，Y2)｝

〔，<パレット番号>〔，｛B ｜ BF｝〕〔，<ラインスタイル>〕

LINE文を書く時の注意は，次のとおりです．

・座標 (X_1, Y_1) から (X_2, Y_2) までに，パレット番号で指定された色の線を書きます．

・座標 (X_1, Y_1) を省略すると，直前のLINE文，CIRCLE文，PAINT文など座標を扱った文の座標がとられます．よって，左の文の意味は，右のようになります．

```
{ LINE (0, 0)-(1, 1)        ⇒ { LINE (0, 0)-(1, 1)
{ LINE       -(2, 2)           { LINE (1, 1)-(2, 2)
```

・LINE文のパラメータに「B」を指定すると，座標 (X_1, Y_1)，(X_2, Y_2) を相対する頂点とする四角を表示します．

例

LINE（0，0）－（10，10），，B

また，「BF」を指定すると，その四角の中を指定されたパレット番号の色で塗りつぶします．

・ラインスタイルを指定することによって，破線や一点鎖線を簡単に引くことができます．詳しくはリファレンスマニュアルを参照して下さい．

⑥ CIRCLE文

円または楕円を描きます．

書き方

CIRCLE ｜（X，Y）｜，〈半径〉〔，パレット番号〉〕
　　　　｜STEP（X，Y）｜
〔，〈開始角度〉〕〔，〈終了角度〉〕〔，比率〕

CIRCLE文を書く時の注意は，次のとおりです．

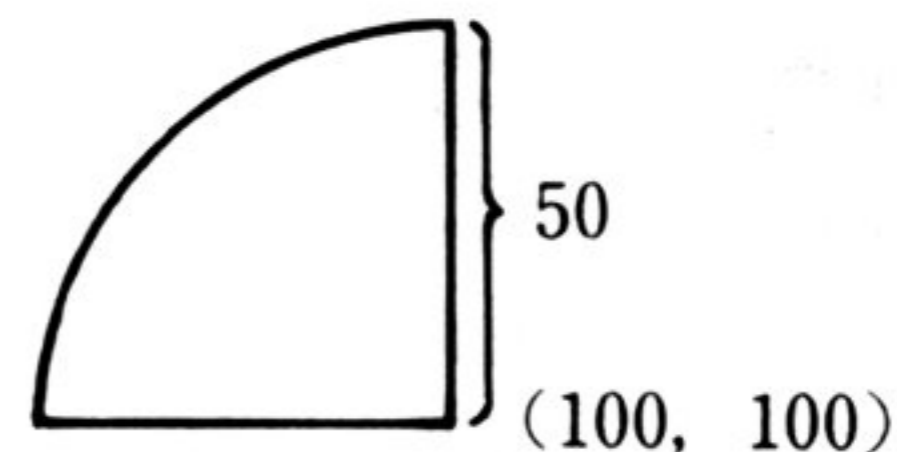
・座標（X，Y）を中心として，指定された半径の円を，指定されたパレット番号の色で書きます．

・座標としてSTEP（X，Y）と書くと，直前のLINE文，CIRCLE文，PAINTなど座標を扱った文の座標からの相対座標がとられます．つまり次の文は同じ意味です．

｛CIRCLE　（100，100），10　　　　⇨　｛CIRCLE（100，100），10
｛CIRCLE　STEP（10，10），10　　　　　｛CIRCLE（110，110），10

・開始角度，終了角度をラジアンで指定すると，円弧を書くことができます．このとき，開始角度，終了角度が負であれば，その絶対値がとられたうえ中心からの半径が描かれますので，扇形を描くことができます．

　例えば，次の命令は図の扇形を描きます．

CIRCLE（100，100），50，，－1.57075，－3.1415

・比率を指定すると楕円を描くことができます．詳しくは，リファレンスマニュアルを参照して下さい．

⑦ CONSOLE文

テキスト画面の各種のモードを指定するためのものです．

書き方

CONSOLE　〔〈スクロール開始行〉〕，〔スクロール行数〉〕
〔，〈ファンクションキー表示スイッチ〉〕
〔，〈カラー／白黒スイッチ〉〕

CONSOLE文を書く時の注意は，次のとおりです．

・ファンクションキー表示スイッチを0にすると，画面最下行に表示されているファンクションキーの文字列を消すことができます．1に戻すと元通り表示します．
例　CONSOLE，，0
・カラー／白黒スイッチを1にすると，テキスト画面がカラーモードになります．0を指定すると白黒モードになります．カラーモードであるか白黒モードであるかによって，COLOR文の実行結果がかわります．
・スクロール開始行とスクロール行数によって，テキスト画面の表示範囲が限定できます．詳しくは，リファレンスマニュアルを参照して下さい．

⑧　COLOR文
テキスト画面の各部の色や，パレットの色を指定します．

書き方

(1)　COLOR　〔〈ファンクションコード〉〕〔，〈バックグランドカラー〉〕
〔，〈ボーダーカラー〉〕〔，フォアグランドカラー〉〕
(2)　COLOR＝（〈パレット番号〉，〈カラーコード〉）
(3)　COLOR@$(X_1, Y_1)-(X_2, Y_2)$〔，〈ファンクションコード〉〕

COLOR文を書く時の注意は次のとおりです．

・書き方(1)の場合，CONSOLE文でテキスト画面がカラーモードになっていれば，テキスト画面に表示する文字に，ファンクションコードで指定した色がつきます．

ファンクションコード（カラーコード）	色
0	黒
1	青
2	赤
3	紫
4	緑
5	水色
6	黄色
7	白

・書き方(2)の場合，各パレット番号にカラーコードで対応する色を指定します．一つの色を複数個のパレット番号に対応させることもできます．

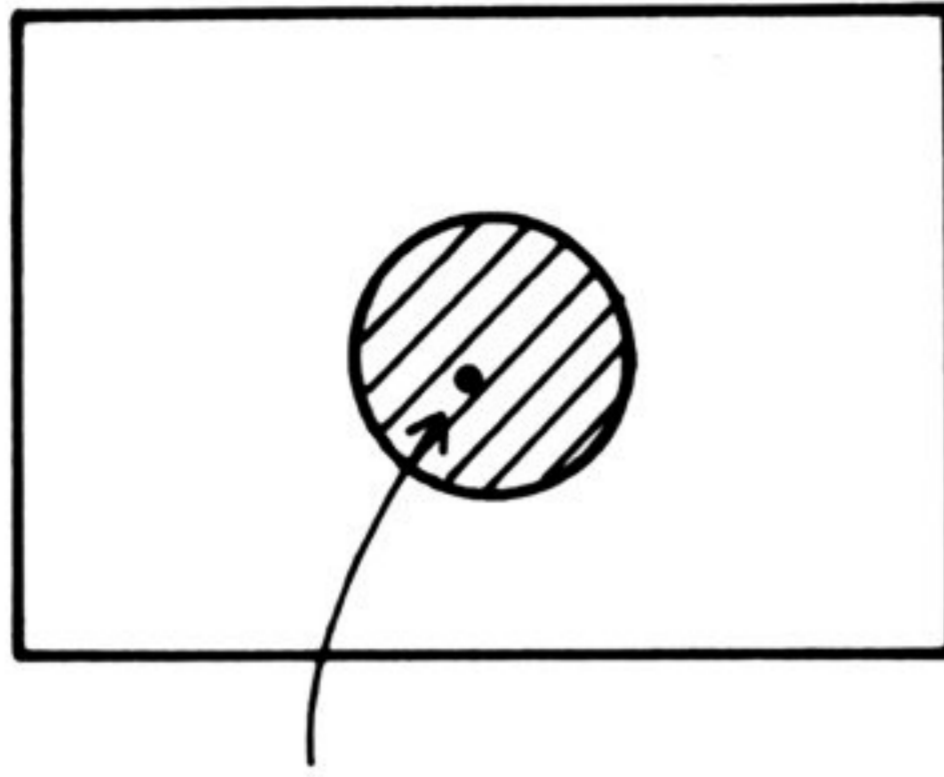
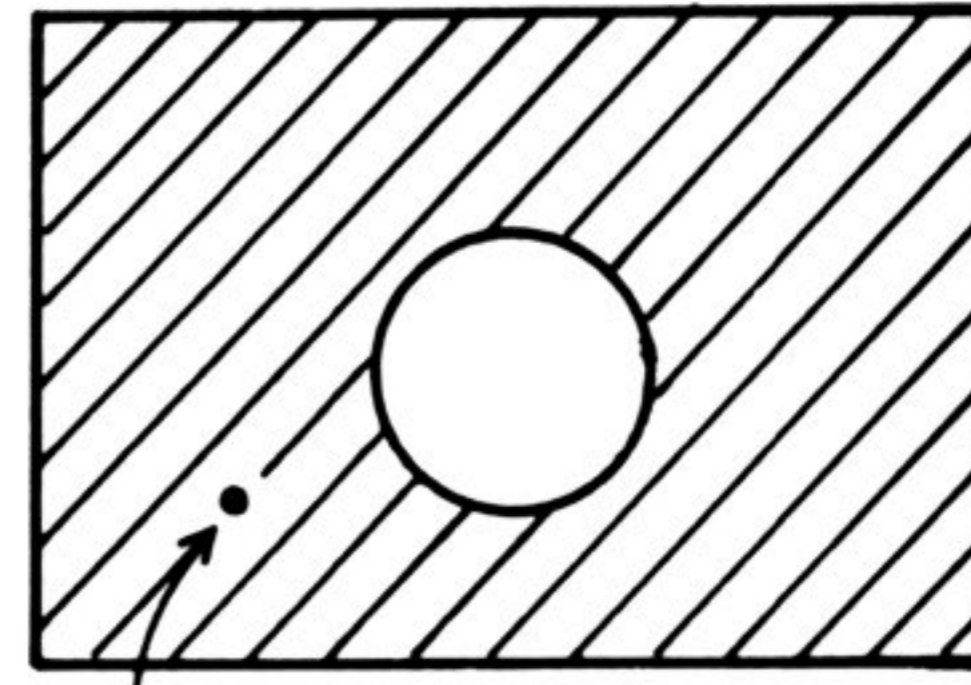
・書き方(3)の場合，テキスト画面の2点の座標で描かれる四角でかこまれた中に表示されている文字を，ファンクションコードで指定した色にすることができます．

(9) PAINT文

指定された色のわくで囲まれた領域を，指定された色で塗ります．

PAINT文を書く時の注意は，次のとおりです．

・境界色で分けられた領域のうち，座標（X，Y）を含む領域を指定された色で塗りつぶします．

例

・画面が境界色で２つに分けられていないと，画面全体が塗りつぶされます．

・座標にSTEPを指定すると相対座標になります．

・指定色，境界色ともパレット番号で指定します．

・PAINT文で中間色を塗ることもできます．詳しくは，リファレンスマニュアルを参照して下さい．

⑩　SIN関数

正弦値（サイン）を与えます．

書き方

SIN（<数式>）

SIN関数を書く時の注意は，次のとおりです．

・数式はラジアン（０～２π）で指定します．

⑪　COS関数

余弦値（コサイン）を与えます．

書き方

COS（<数式>）

COS関数を書く時の注意は，次のとおりです．

・数式はラジアン（０～２π）で指定します．

付　録

本章では，プログラムを保存しておくディスクファイルの扱い方など，説明の足りなかった部分を補足します．

第1節　フロッピィディスクの扱い

ここでは，プログラムを保存するフロッピィディスクの扱い方について学びましょう．

1 ファイル一覧表の表示／印刷

3章で，プログラムをフロッピィディスクに格納したり，格納されているプログラムを主記憶装置に読み出す方法，つまり，SAVE文，LOAD文の使い方について説明しました．

ここでは，フロッピィディスクに格納されているファイルの一覧表について説明しましょう．

・FILES文およびLFILES文

フロッピィディスクのファイル一覧を画面に表示するには，FILESを使います．フロッピィディスクをセットして，次の様にキー入力して下さい．

FILES ⏎

画面にファイルの一覧表が出力されましたね．

また，ファイル一覧表をプリンタに印字するには，LFILES文を使います．

LFILES ⏎

2 ファイル名の変更

フロッピィディスク上のファイルの名前の変更が必要になることがあります．その場合，NAME文で簡単に変更することができます．「TEST1」という名前を「KEISAN」という名前にかえる場合，次の様にキー入力します．

NAME "TEST 1" AS "KEISAN" ⏎

3 ファイルの削除

フロッピィディスク上のファイルで不要になったものを削除する場合，KILL文によって行います．「TEST2」というプログラムが不要になったときは，次の様にキー入力します．

KILL "TEST2" ⏎

4 ファイルの保護

フロッピィディスク上のファイルをSET文を使って保護することができます．その理由は，ファイルをKILL文で勝手に消去したり，ファイル名を変更したり，ファイル中に別の関係のないデータを書き込んだりして，トラブルを起さないようにするためです．1枚のフロッピイ全体を保護する場合次の様に指定します．

SET 1, "P" ⏎

ある特定のファイルだけ保護したい場合，例えば，「DATA1」ファイルを保護する場合は，次の様に指定します．

SET "1：DATA1", "P"

第2節　よく使う関数

BASICには，コマンド文と関数があることはすでに説明しました．関数のうち，TAB関数については2章で，EOF関数については3章で，LEFT$，RIGHT$の各関数については5章で，さらに，SIN，COS，WINDOW，VIEWの各関数については6章で学んできました．

ここでは，上記以外の関数で特によく使う関数について学びましょう．

1　日付や時間を知る（DATE$関数，TIME$関数）

プログラムの実行に当って，日付や時間が欲しいことがあります．日付を得るためにはDATE$関数，時間を得るためにはTIME$関数を使います．次のようにやってみましょう．

? date$ ⏎

? time$ ⏎

本体にはカレンダ時計が内蔵されています．この時計は電源をOFFにしても電池で動いています．ですから，一度正しい時刻をセットすればよいわけですが，もし日付や時刻が違っている場合，あらためて正しい日付や時刻がセットできます．

date$＝“82／10／10” ⏎

time$＝“10：00：00” ⏎

2　文字列である数字を数値に変える（VAL関数）

数字は，文字列として与えると，演算に使えません．ですから，次の様な命令は許されません．

A$＝“100”　　　：B＝A$＋10

この文字列中の数字を演算できる形に変換するのが，VAL関数です．上記の計算は，次のようにすればできます．

A$＝“100”　　：A＝VAL（A$）　　：B＝A＋10

もし，〈文字列〉に，＋，−，＆以外の文字が含まれていると結果は0になります．
文字列中のスペースは無視されます．

おわりに

さて，本書を読み終って，N_{88}-日本語BASIC (86) 言語を使ったプログラミングにかなりの自信を得たことと思います．

PC-9800シリーズパーソナルコンピュータは,日本語,グラフィック,回線による通信など驚くほどすばらしい機能を持っています．そして，これらの機能はすべてN_{88}-日本語BASIC (86) により容易に利用することができます．本書で学んだ知識をユーザーズマニュアル，リファレンスマニュアルによりさらに応用発展させ，より深い知識を修得されることを望んでやみません．